Name: Brownwell Ehimare

Date:29/02/2024

Course: Computer Science

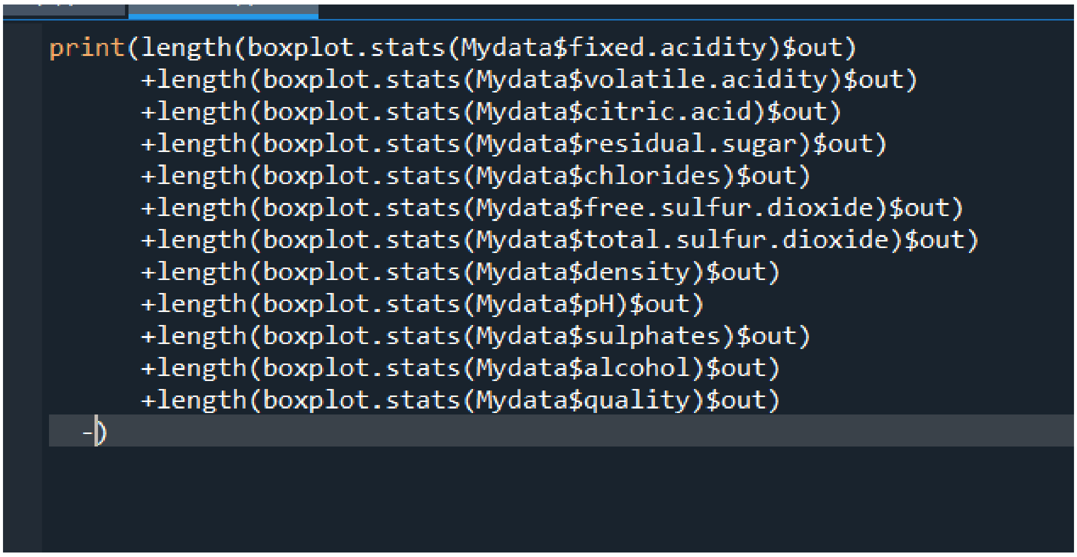
Coursework 1

Subtask 1:

Before conducting the k-means, perform the following pre-processing tasks: scaling and outliers detection/removal and briefly justify your answer. (Suggestion: the order of scaling and outliers removal is important. The outlier removal topic is not covered in tutorials, so you need to explore it yourself). Obviously, you can implement this clustering task without exploring this “outlier ”component, however, you will not be awarded the allocated marks for this comp

1)Use box-plot for outlier Detection and Diagram Analysis:

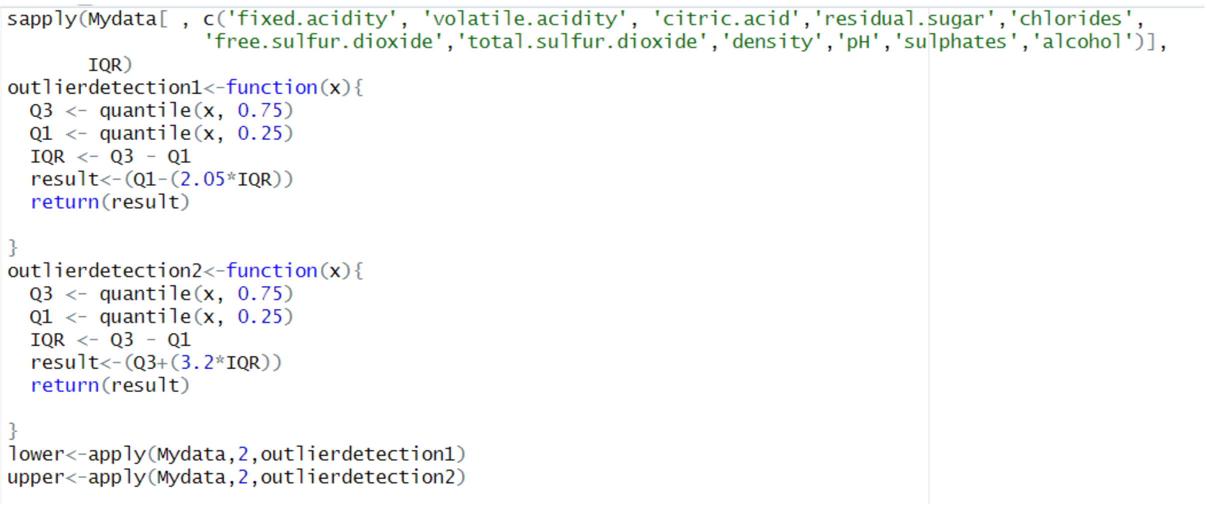
538 outlier detected



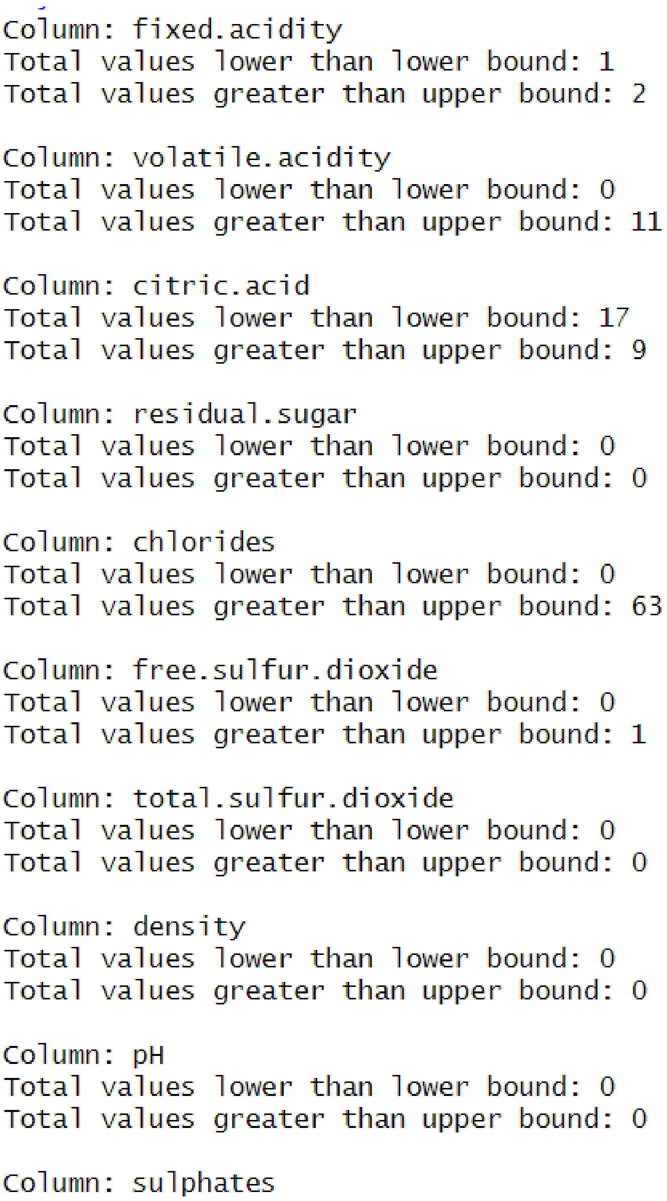
```
print(length(boxplot.stats(Mydata$fixed.acidity)$out)
      +length(boxplot.stats(Mydata$volatile.acidity)$out)
      +length(boxplot.stats(Mydata$citric.acid)$out)
      +length(boxplot.stats(Mydata$residual.sugar)$out)
      +length(boxplot.stats(Mydata$chlorides)$out)
      +length(boxplot.stats(Mydata$free.sulfur.dioxide)$out)
      +length(boxplot.stats(Mydata$total.sulfur.dioxide)$out)
      +length(boxplot.stats(Mydata$density)$out)
      +length(boxplot.stats(Mydata$pH)$out)
      +length(boxplot.stats(Mydata$sulphates)$out)
      +length(boxplot.stats(Mydata$alcohol)$out)
      +length(boxplot.stats(Mydata$quality)$out)
  )
```

After doing Experiment with scaling before outlier detection it severely affected the data by increasing the minimum value of the data higher than normal,

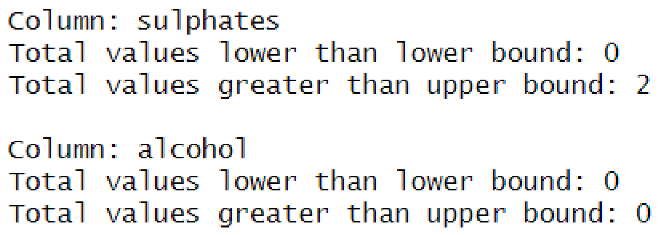
I used outlier detection before scaling, to adequately remove the outlier and not make them merge and severely affect other data



```
sapply(Mydata[ , c('fixed.acidity', 'volatile.acidity', 'citric.acid','residual.sugar','chlorides',
                  'free.sulfur.dioxide','total.sulfur.dioxide','density','pH','sulphates','alcohol')],
       IQR)
outlierdetection1<-function(x){
  Q3 <- quantile(x, 0.75)
  Q1 <- quantile(x, 0.25)
  IQR <- Q3 - Q1
  result<-(Q1-(2.05*IQR))
  return(result)

}
outlierdetection2<-function(x){
  Q3 <- quantile(x, 0.75)
  Q1 <- quantile(x, 0.25)
  IQR <- Q3 - Q1
  result<-(Q3+(3.2*IQR))
  return(result)

}
lower<-apply(Mydata,2,outlierdetection1)
upper<-apply(Mydata,2,outlierdetection2)
```

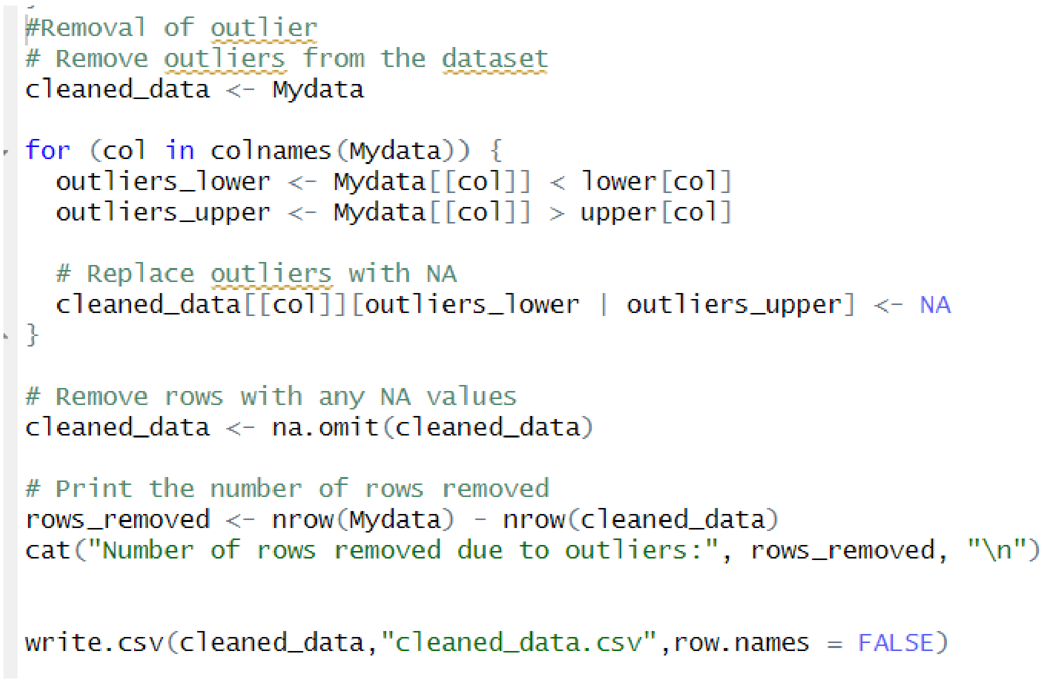


```
Column: fixed.acidity
Total values lower than lower bound: 1
Total values greater than upper bound: 2

Column: volatile.acidity
Total values lower than lower bound: 0
Total values greater than upper bound: 11

Column: citric.acid
Total values lower than lower bound: 17
Total values greater than upper bound: 9

Column: residual.sugar
Total values lower than lower bound: 0
Total values greater than upper bound: 0

Column: chlorides
Total values lower than lower bound: 0
Total values greater than upper bound: 63

Column: free.sulfur.dioxide
Total values lower than lower bound: 0
Total values greater than upper bound: 1

Column: total.sulfur.dioxide
Total values lower than lower bound: 0
Total values greater than upper bound: 0

Column: density
Total values lower than lower bound: 0
Total values greater than upper bound: 0

Column: pH
Total values lower than lower bound: 0
Total values greater than upper bound: 0

Column: sulphates
```



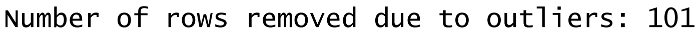
```
Column: sulphates
Total values lower than lower bound: 0
Total values greater than upper bound: 2

Column: alcohol
Total values lower than lower bound: 0
Total values greater than upper bound: 0
```

code in removing the outlier:



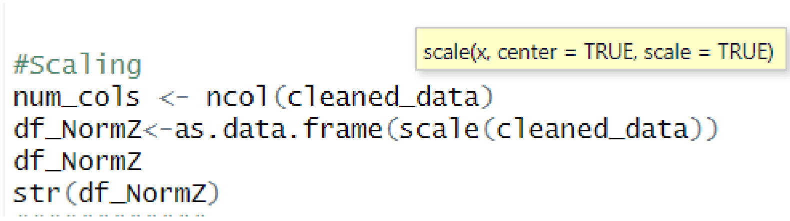
```
#Removal of outlier
# Remove outliers from the dataset
cleaned_data <- Mydata

for (col in colnames(Mydata)) {
  outliers_lower <- Mydata[[col]] < lower[col]
  outliers_upper <- Mydata[[col]] > upper[col]

  # Replace outliers with NA
  cleaned_data[[col]][outliers_lower | outliers_upper] <- NA
}

# Remove rows with any NA values
cleaned_data <- na.omit(cleaned_data)

# Print the number of rows removed
rows_removed <- nrow(Mydata) - nrow(cleaned_data)
cat("Number of rows removed due to outliers:", rows_removed, "\n")


write.csv(cleaned_data,"cleaned_data.csv",row.names = FALSE)
```

Output:



```
Number of rows removed due to outliers: 101
```

Scaling the data(code):

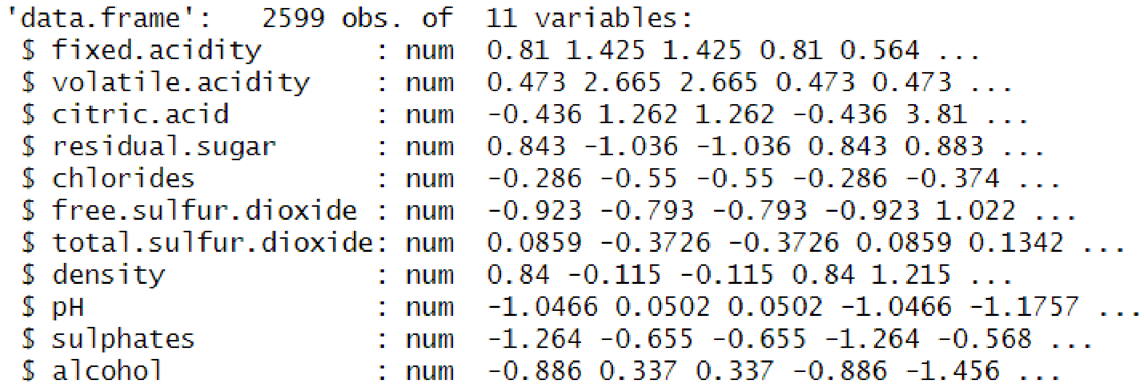


```
#Scaling
num_cols <- ncol(cleaned_data)
df_NormZ<-as.data.frame(scale(cleaned_data))
df_NormZ
str(df_NormZ)
```

scale(x, center = TRUE, scale = TRUE)

Scaling the data(output):

As you can see from the diagram below that 101 outliers and NA values was removed from the data making the data reduced to “2599”

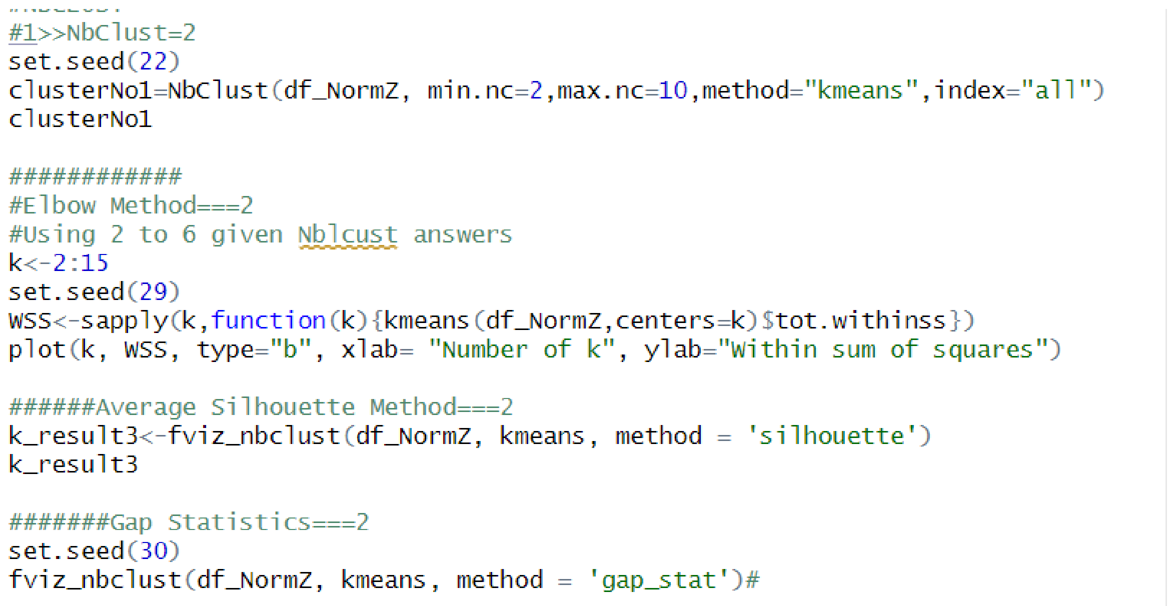


```
'data.frame':   2599 obs. of  11 variables:
 $ fixed.acidity       : num  0.81 1.425 1.425 0.81 0.564 ...
 $ volatile.acidity    : num  0.473 2.665 2.665 0.473 0.473 ...
 $ citric.acid         : num  -0.436 1.262 1.262 -0.436 3.81 ...
 $ residual.sugar      : num  0.843 -1.036 -1.036 0.843 0.883 ...
 $ chlorides           : num  -0.286 -0.55 -0.55 -0.286 -0.374 ...
 $ free.sulfur.dioxide : num  -0.923 -0.793 -0.793 -0.923 1.022 ...
 $ total.sulfur.dioxide: num  0.0859 -0.3726 -0.3726 0.0859 0.1342 ...
 $ density             : num  0.84 -0.115 -0.115 0.84 1.215 ...
 $ pH                  : num  -1.0466 0.0502 0.0502 -1.0466 -1.1757 ...
 $ sulphates           : num  -1.264 -0.655 -0.655 -1.264 -0.568 ...
 $ alcohol             : num  -0.886 0.337 0.337 -0.886 -1.456 ...
```

<u>Subtask 2:</u>

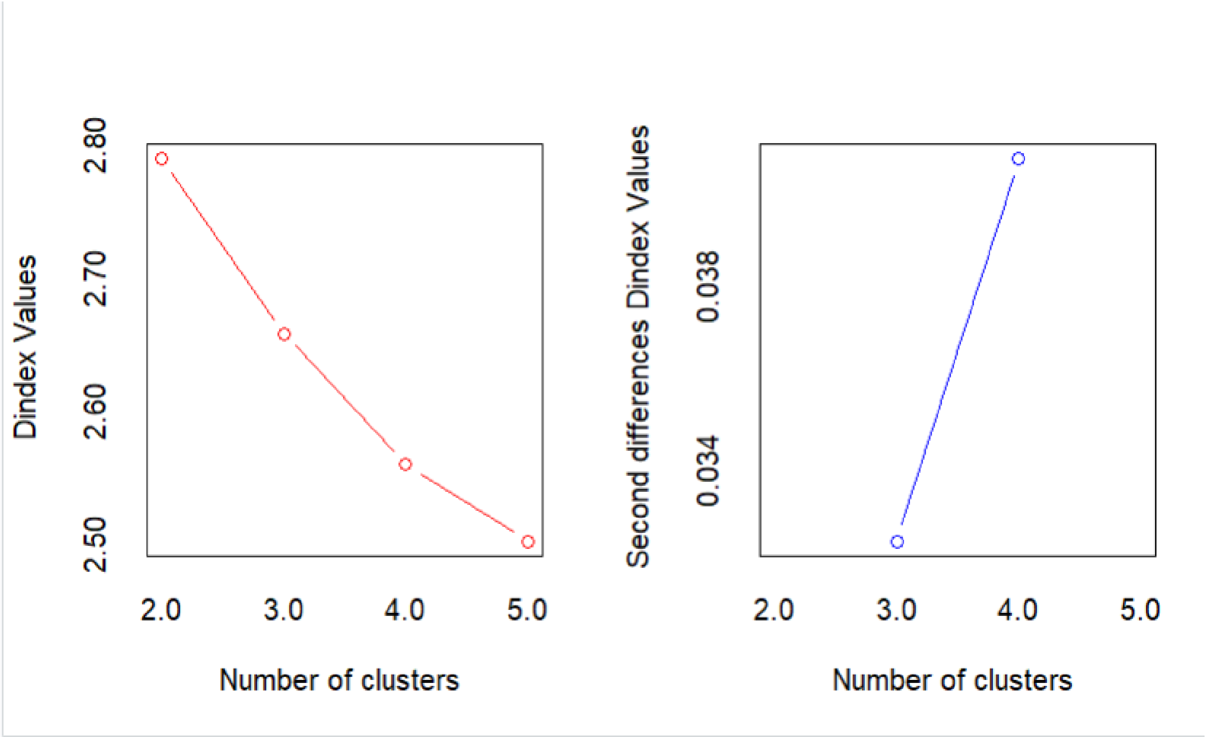
Determine the number of cluster centres by showing all necessary steps/methods via “automated “tools (1.5 mark for each one of these “automated” tools)

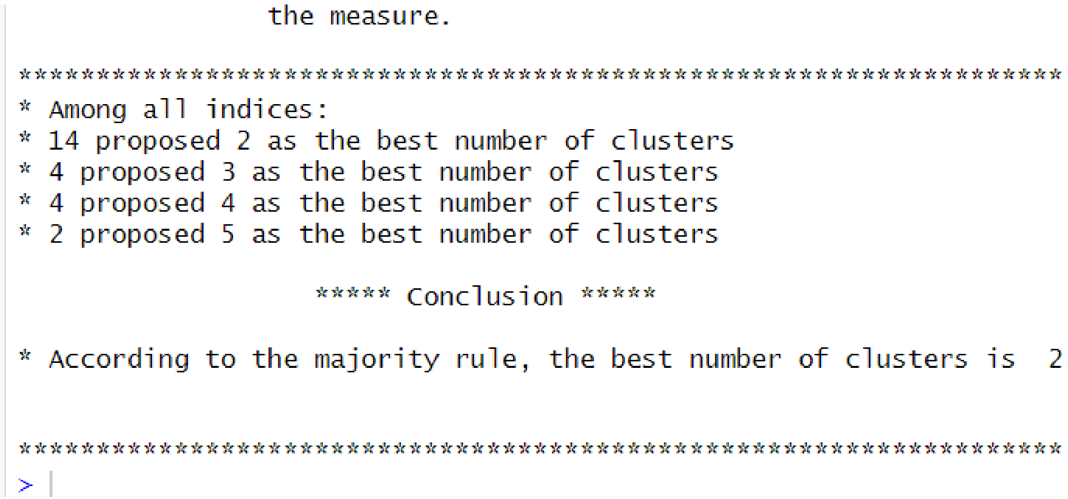
The four automated tool: Nbclust . Elbow Method ,Average Silhouette Method ,Gap Statistics



```
#1>>NbClust=2
set.seed(22)
clusterNo1=NbClust(df_NormZ, min.nc=2,max.nc=10,method="kmeans",index="all")
clusterNo1

############
#Elbow Method===2
#Using 2 to 6 given Nblcust answers
k<-2:15
set.seed(29)
WSS<-sapply(k,function(k){kmeans(df_NormZ,centers=k)$tot.withinss})
plot(k, WSS, type="b", xlab= "Number of k", ylab="Within sum of squares")

######Average Silhouette Method===2
k_result3<-fviz_nbclust(df_NormZ, kmeans, method = 'silhouette')
k_result3

#######Gap Statistics===2
set.seed(30)
fviz_nbclust(df_NormZ, kmeans, method = 'gap_stat')#
```

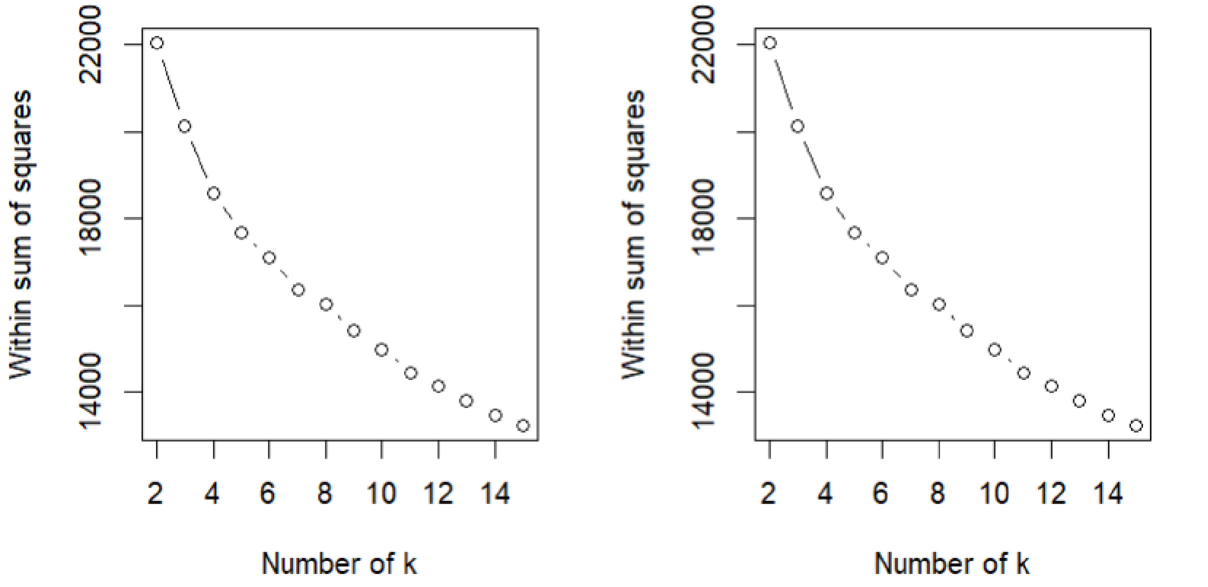
NBCLUST(OUTPUT):2



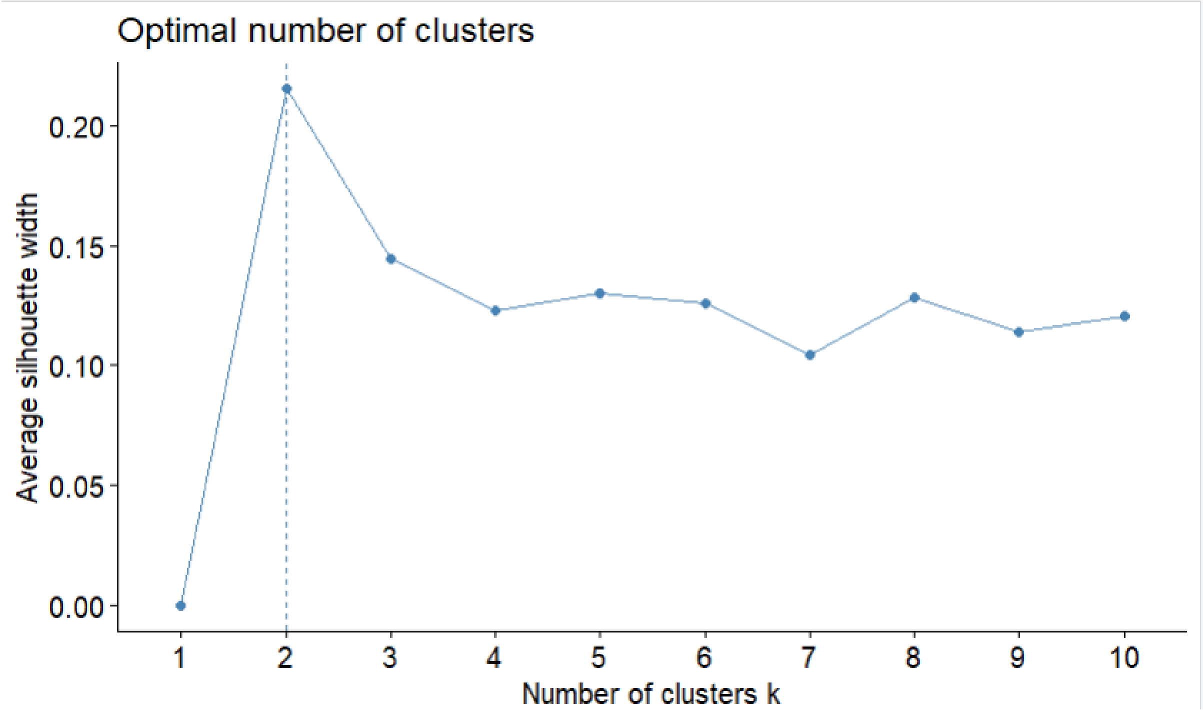


```
                the measure.

*******************************************************************
* Among all indices:
* 14 proposed 2 as the best number of clusters
* 4 proposed 3 as the best number of clusters
* 4 proposed 4 as the best number of clusters
* 2 proposed 5 as the best number of clusters

                   ***** Conclusion *****

* According to the majority rule, the best number of clusters is  2


*******************************************************************
> |
```

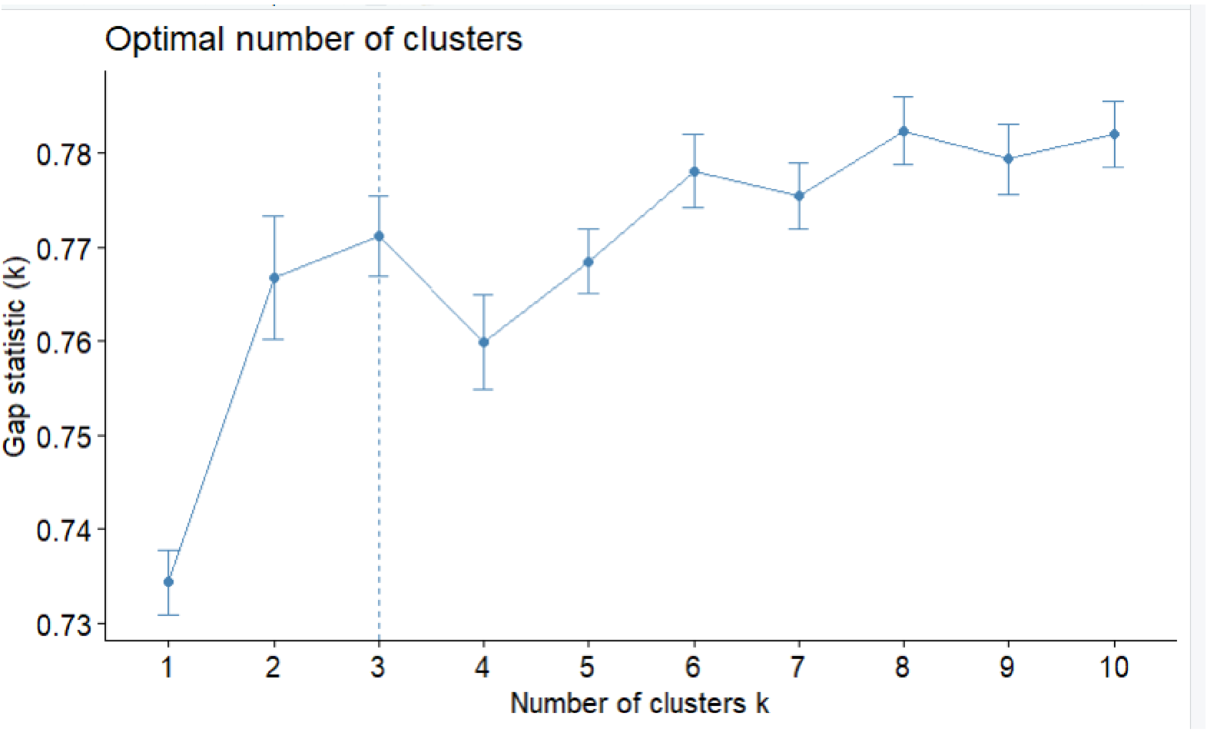
ELBOW METHOD(OUTPUT):2



AVERAGE SHILEOUTTE PLOT(OUTPUT): 2



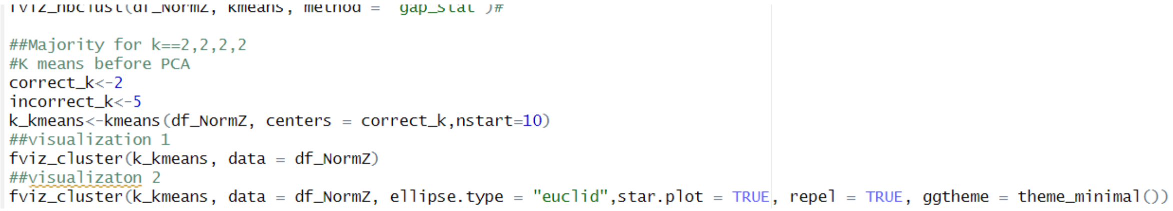
GAT STATISTICS(OUTPUT):3



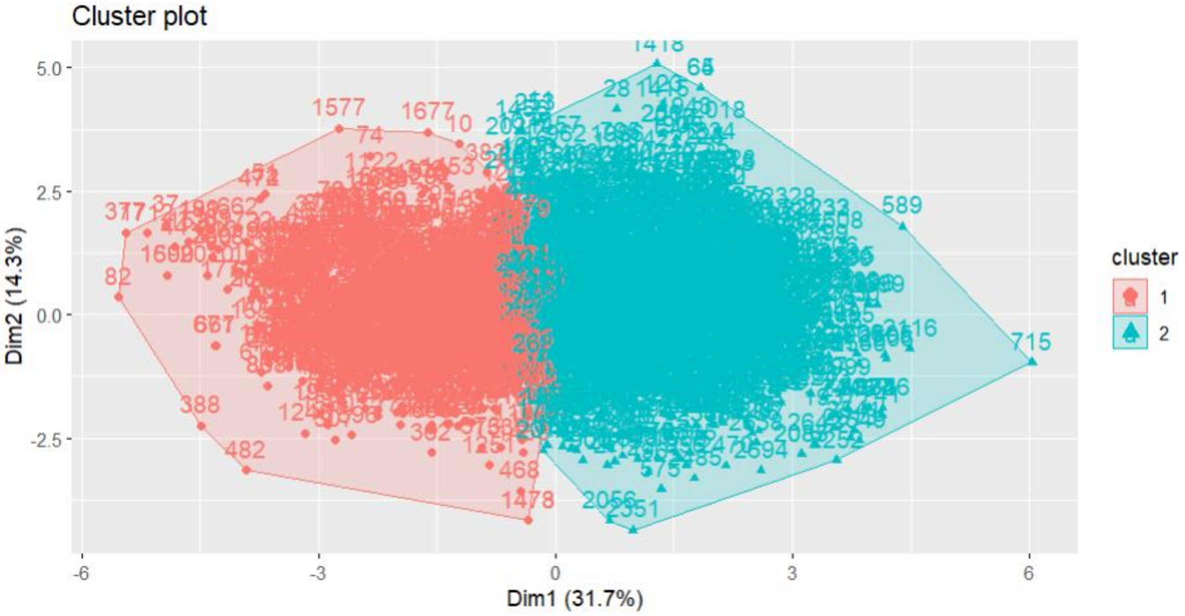
This gives a majority decision of ” k=2”

Subtask 3:

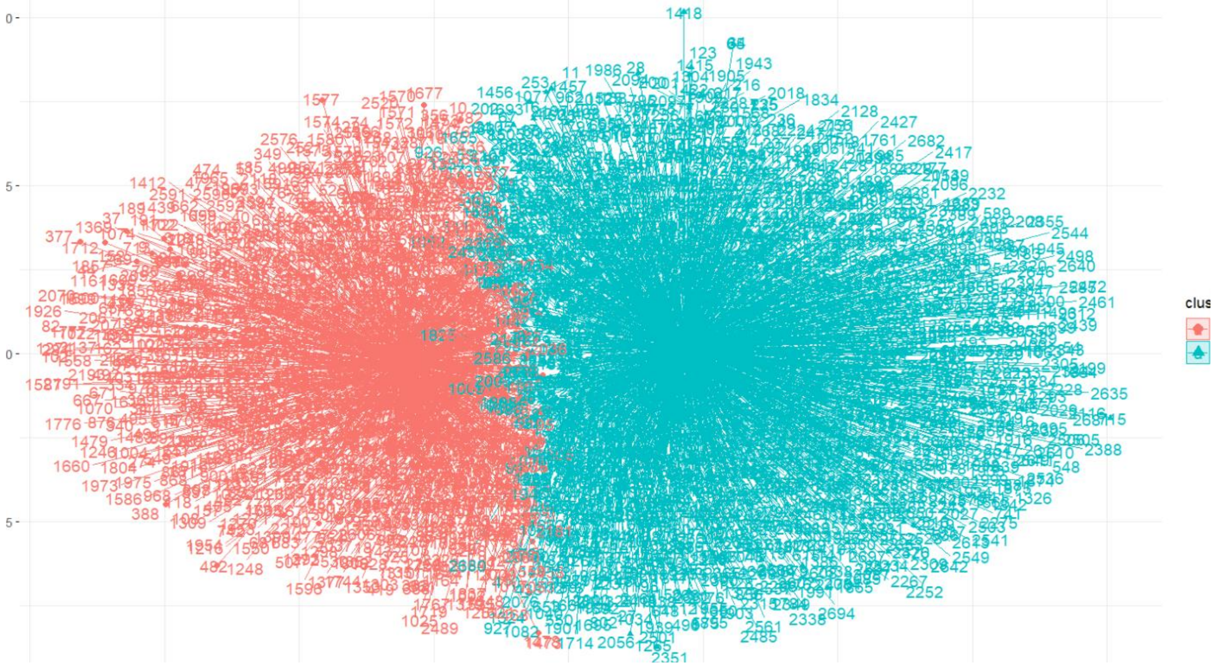
K-means analysis for the chosen k (all attributes used) and show all requested outputs, Show the silhouette plot (3 marks) and provide related discussion on this output, following this Kmeans attempt:



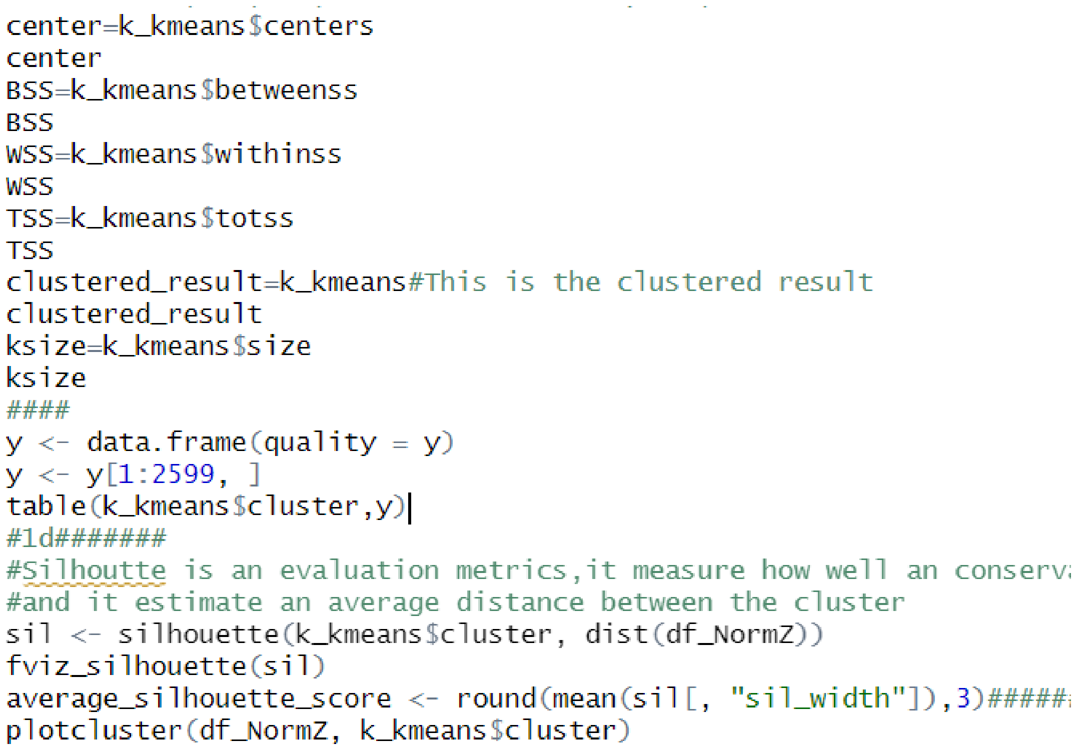
```
fviz_nbclust(df_NormZ, kmeans, method = "gap_stat")#

##Majority for k==2,2,2,2
#K means before PCA
correct_k<-2
incorrect_k<-5
k_kmeans<-kmeans(df_NormZ, centers = correct_k,nstart=10)
##visualization 1
fviz_cluster(k_kmeans, data = df_NormZ)
##visualizaton 2
fviz_cluster(k_kmeans, data = df_NormZ, ellipse.type = "euclid",star.plot = TRUE, repel = TRUE, ggtheme = theme_minimal())
```



Plot 2;

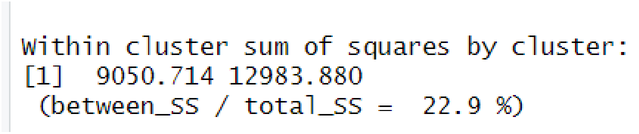


Silhouette plot(code):

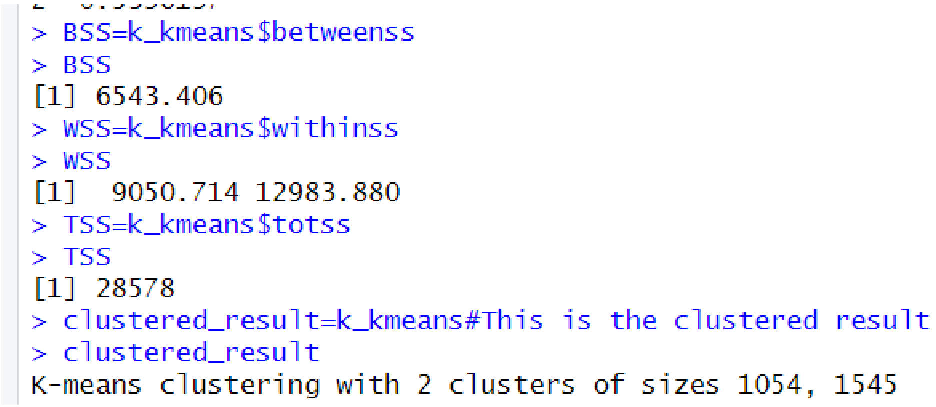


```
center=k_kmeans$centers
center
BSS=k_kmeans$betweenss
BSS
WSS=k_kmeans$withinss
WSS
TSS=k_kmeans$totss
TSS
clustered_result=k_kmeans#This is the clustered result
clustered_result
ksize=k_kmeans$size
ksize
####
y <- data.frame(quality = y)
y <- y[1:2599, ]
table(k_kmeans$cluster,y)
#1d#######
#Silhoutte is an evaluation metrics,it measure how well an conserv
#and it estimate an average distance between the cluster
sil <- silhouette(k_kmeans$cluster, dist(df_NormZ))
fviz_silhouette(sil)
average_silhouette_score <- round(mean(sil[, "sil_width"]),3)#####
plotcluster(df_NormZ, k_kmeans$cluster)
```

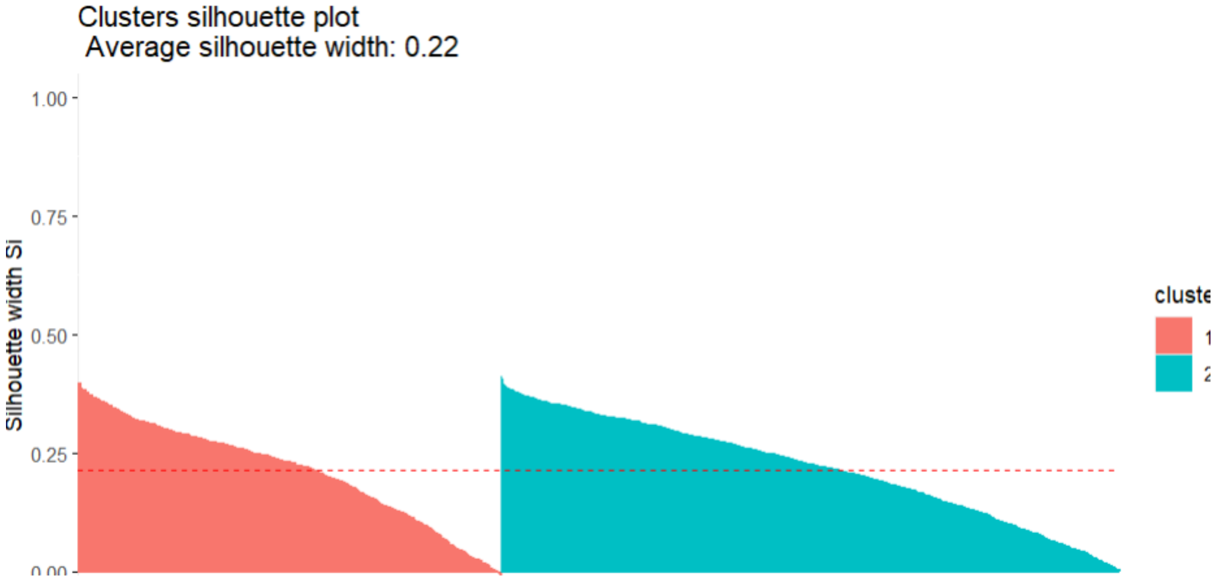
Silhouette plot and others(output):

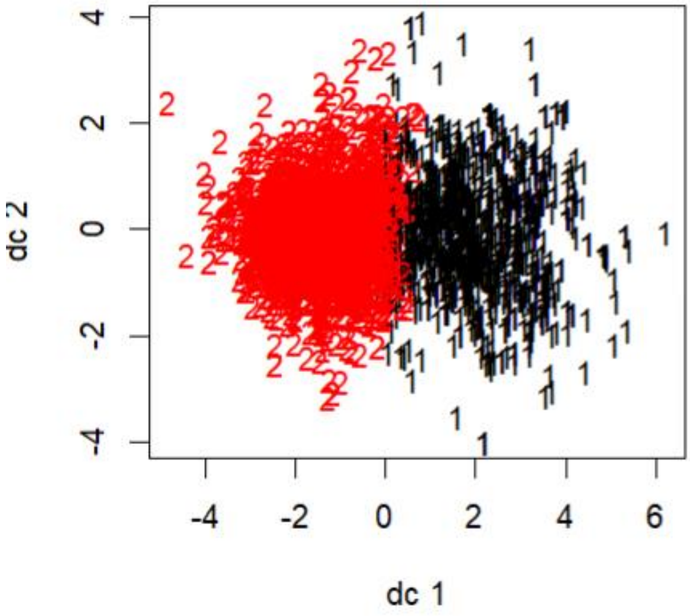


```
Within cluster sum of squares by cluster:
[1]  9050.714 12983.880
 (between_SS / total_SS =  22.9 %)
```



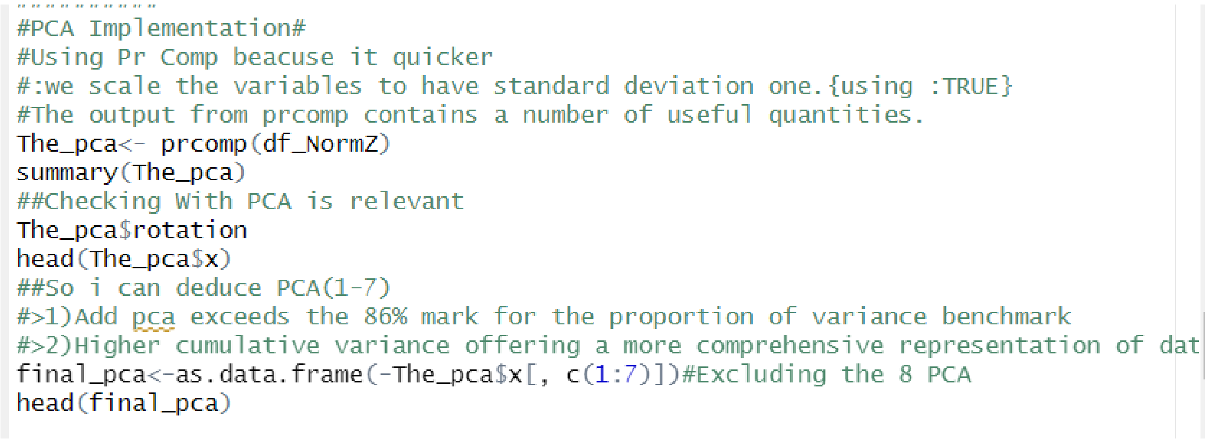
```
> BSS=k_kmeans$betweenss
> BSS
[1] 6543.406
> WSS=k_kmeans$withinss
> WSS
[1]  9050.714 12983.880
> TSS=k_kmeans$totss
> TSS
[1] 28578
> clustered_result=k_kmeans#This is the clustered result
> clustered_result
K-means clustering with 2 clusters of sizes 1054, 1545
```



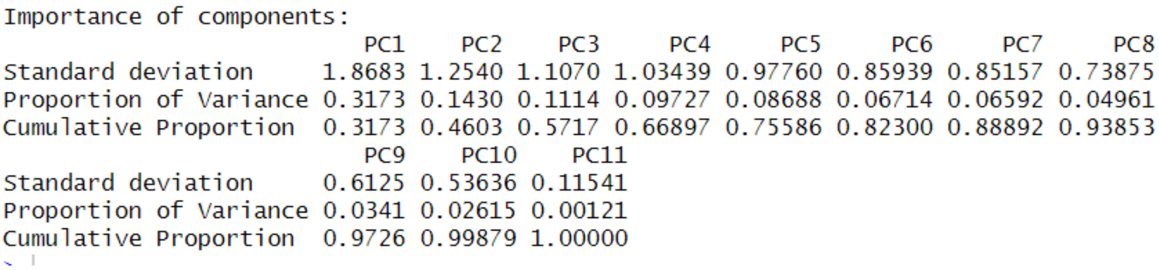


Subtask 4:

Apply a PCA for this wine dataset and show all related R-outputs (4 marks). Create a new dataset with those PCs with a cumulative score at least > 85%, as attributes and provide a discussion for your choice (4 marks).



```
#PCA Implementation#
#Using Pr Comp beacuse it quicker
#:we scale the variables to have standard deviation one.{using :TRUE}
#The output from prcomp contains a number of useful quantities.
The_pca<- prcomp(df_NormZ)
summary(The_pca)
##Checking With PCA is relevant
The_pca$rotation
head(The_pca$x)
##So i can deduce PCA(1-7)
#>1)Add pca exceeds the 86% mark for the proportion of variance benchmark
#>2)Higher cumulative variance offering a more comprehensive representation of dat
final_pca<-as.data.frame(-The_pca$x[, c(1:7)])#Excluding the 8 PCA
head(final_pca)
```



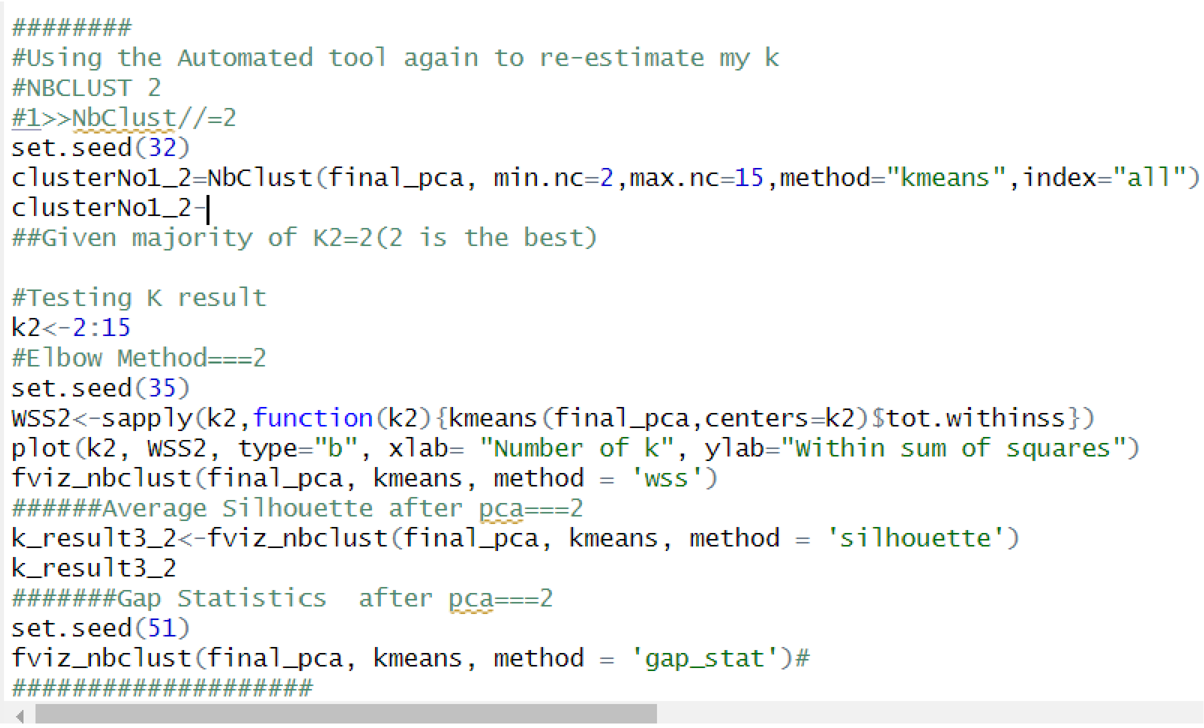
```
Importance of components:
                          PC1    PC2    PC3     PC4     PC5     PC6     PC7     PC8
Standard deviation     1.8683 1.2540 1.1070 1.03439 0.97760 0.85939 0.85157 0.73875
Proportion of Variance 0.3173 0.1430 0.1114 0.09727 0.08688 0.06714 0.06592 0.04961
Cumulative Proportion  0.3173 0.4603 0.5717 0.66897 0.75586 0.82300 0.88892 0.93853
                          PC9    PC10    PC11
Standard deviation     0.6125 0.53636 0.11541
Proportion of Variance 0.0341 0.02615 0.00121
Cumulative Proportion  0.9726 0.99879 1.00000
```

Given the code above I chose the PCA 1 to 7 has they gave a proportion variance of over 86% (0.3173+0.1430+0.1114+0.09727+0.0868+ 0.06714+0.06592===~88%
),There is no need to go beyond those seven has they satisfy the conditions above and they still retain relevant information’s

Subtask 5:

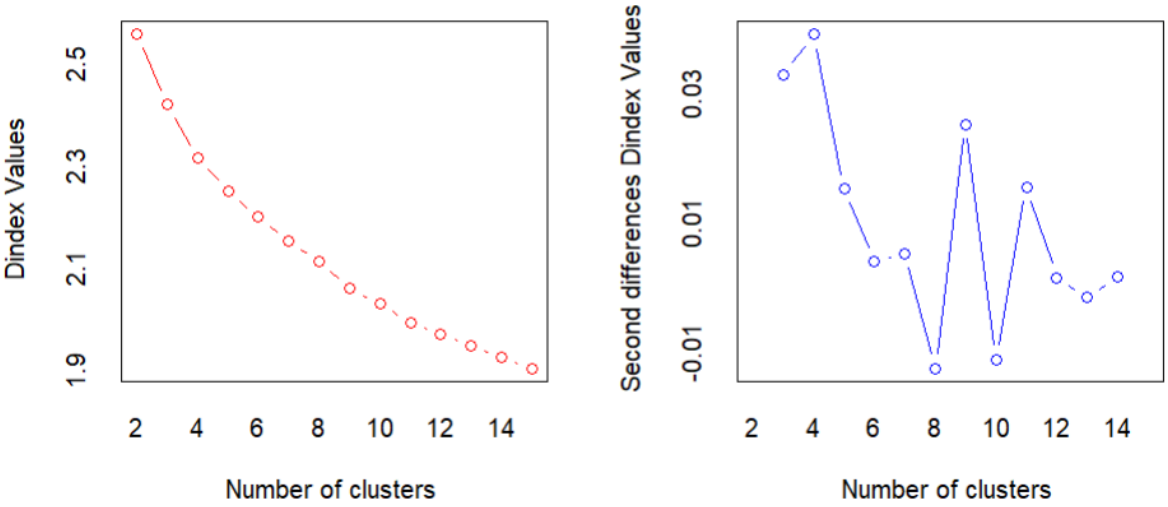
Determine the number of cluster centres by showing all necessary steps/methods via “automated”tools (1.5 mark for each one of these “automated” tools)

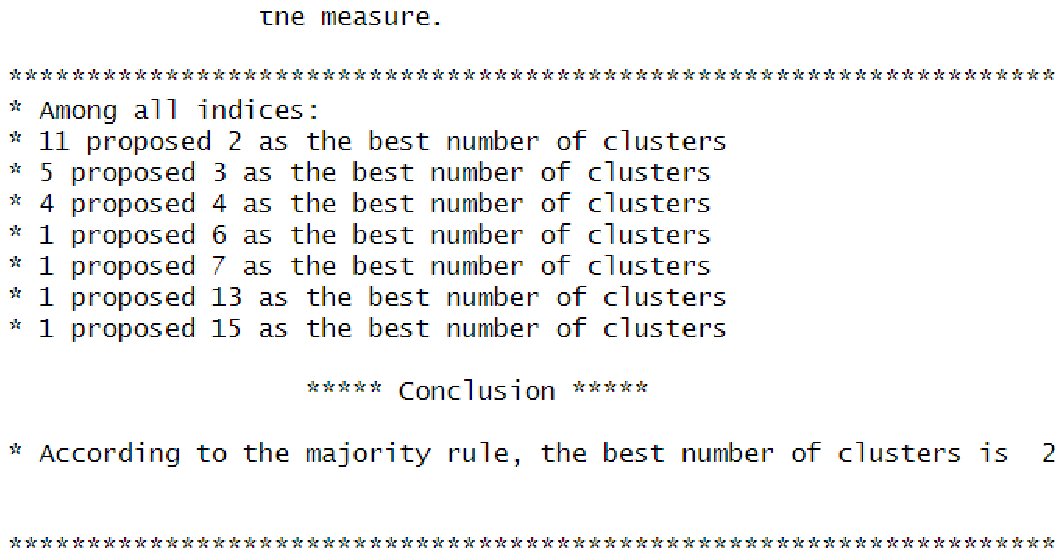
Code Implementation:



```
########
#Using the Automated tool again to re-estimate my k
#NBCLUST 2
#1>>NbClust//=2
set.seed(32)
clusterNo1_2=NbClust(final_pca, min.nc=2,max.nc=15,method="kmeans",index="all")
clusterNo1_2-
##Given majority of K2=2(2 is the best)

#Testing K result
k2<-2:15
#Elbow Method===2
set.seed(35)
WSS2<-sapply(k2,function(k2){kmeans(final_pca,centers=k2)$tot.withinss})
plot(k2, WSS2, type="b", xlab= "Number of k", ylab="Within sum of squares")
fviz_nbclust(final_pca, kmeans, method = 'wss')
######Average Silhouette after pca===2
k_result3_2<-fviz_nbclust(final_pca, kmeans, method = 'silhouette')
k_result3_2
#######Gap Statistics  after pca===2
set.seed(51)
fviz_nbclust(final_pca, kmeans, method = 'gap_stat')#
####################
```

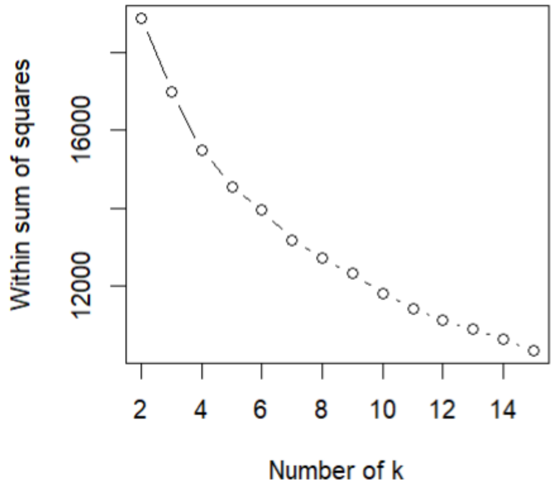
NBCLUST AFTER PCA:

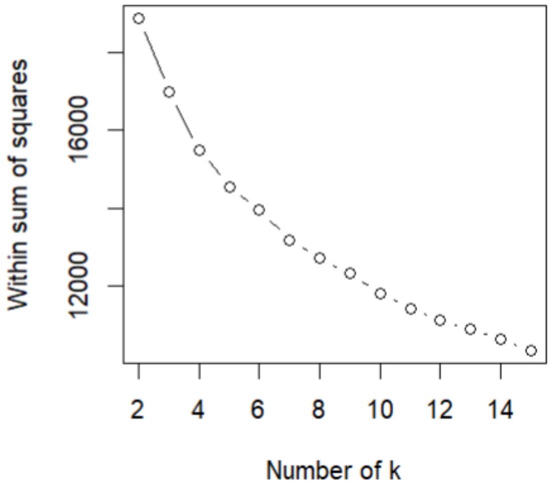




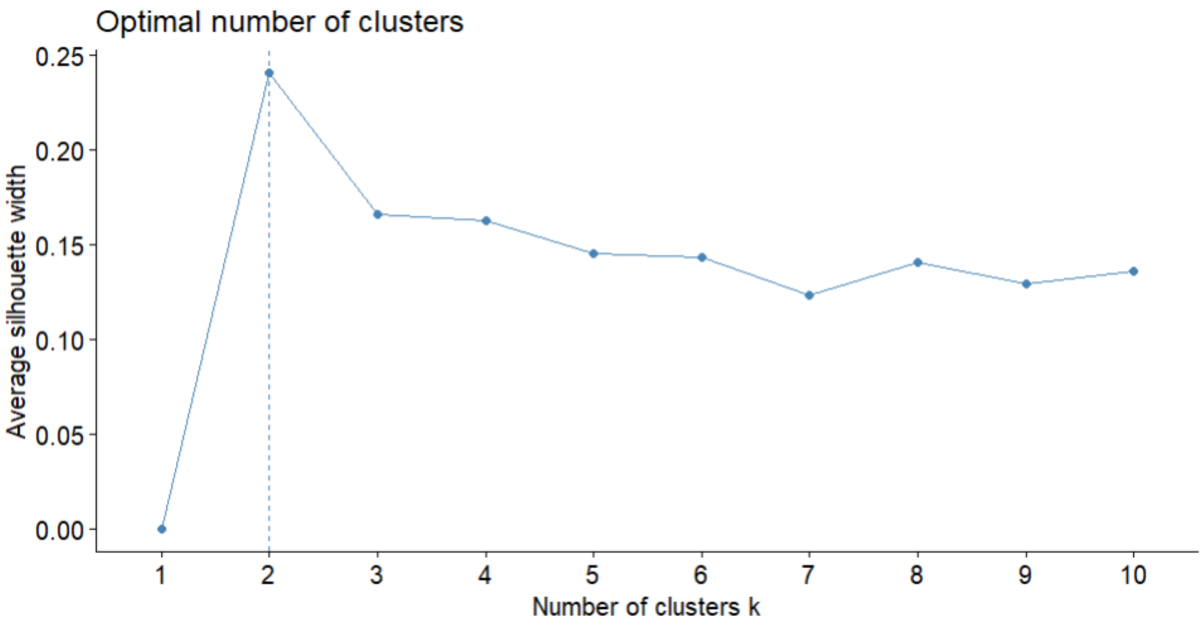
```
                 the measure.

*******************************************************************
* Among all indices:
* 11 proposed 2 as the best number of clusters
* 5 proposed 3 as the best number of clusters
* 4 proposed 4 as the best number of clusters
* 1 proposed 6 as the best number of clusters
* 1 proposed 7 as the best number of clusters
* 1 proposed 13 as the best number of clusters
* 1 proposed 15 as the best number of clusters

                   ***** Conclusion *****

* According to the majority rule, the best number of clusters is  2


*******************************************************************
```

ELBOW METHOD AFTER PCA:

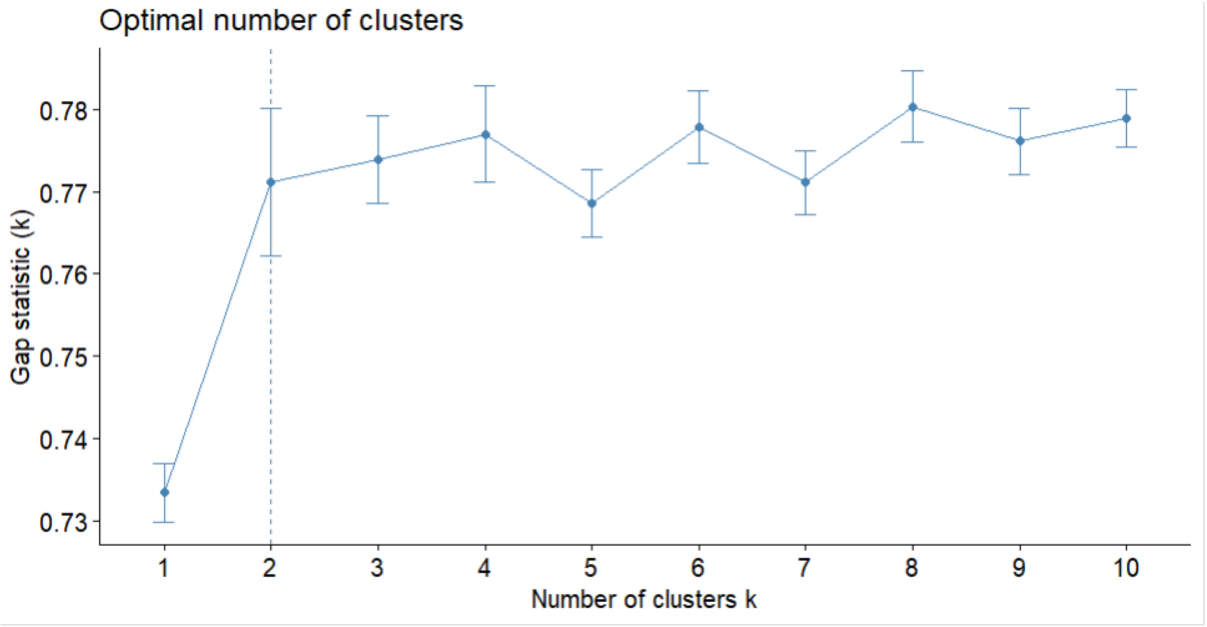




AVERAGE SHILEOUTTUE AFTER PCA:



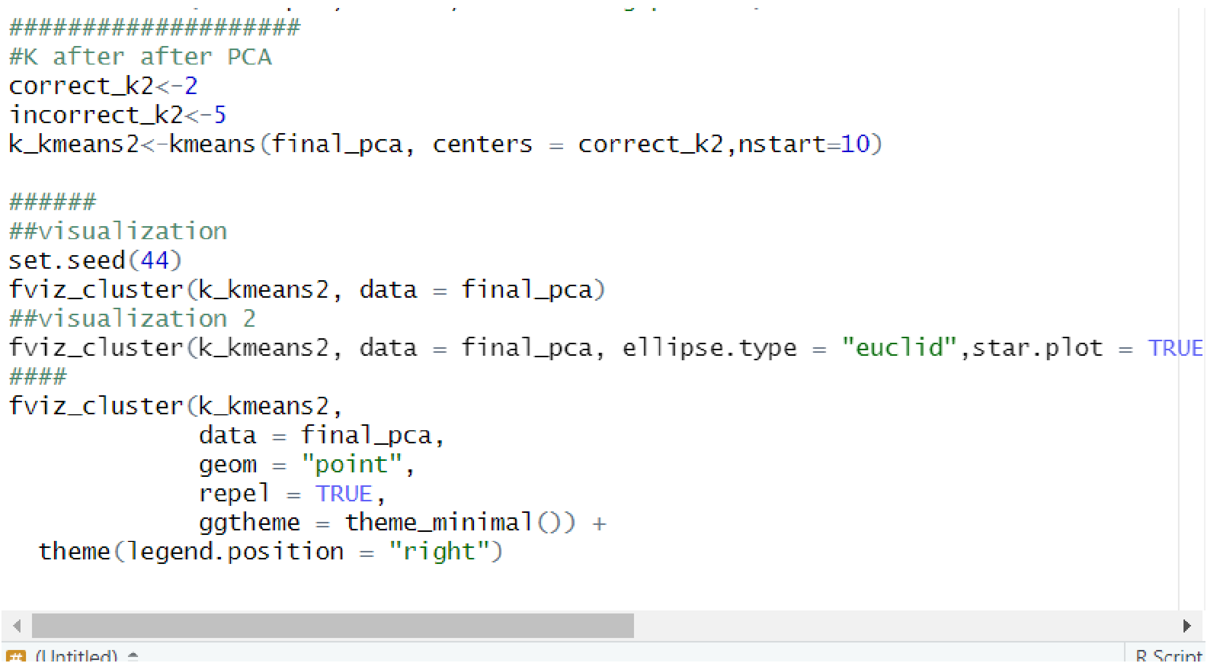
GAP STATISTICS AFTER PCA:



This gives a total majority of kmeans=2 after pca

SubTask 6:

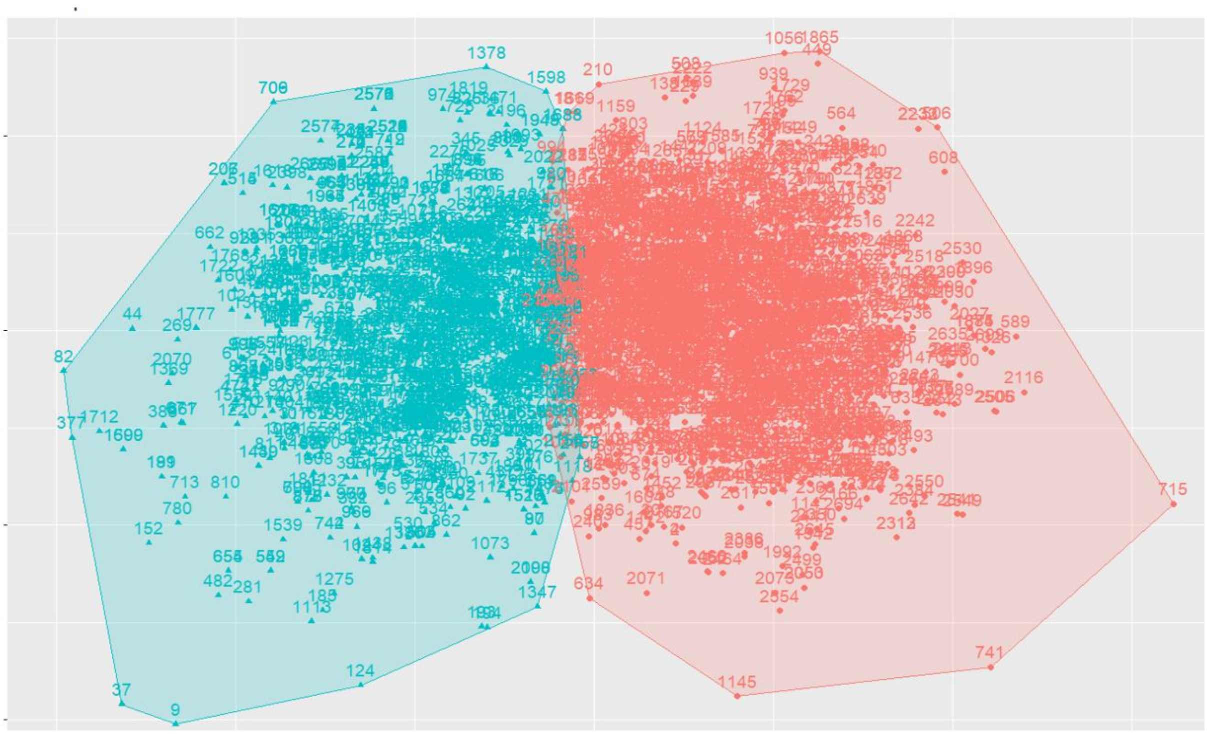
K-means analysis for this “pca”-based dataset for the chosen k and show all requested outputs 58. Show the silhouette plot (3 marks) and provide related discussion on this output, following this“pca-based” Kmeans attempt (2 marks)



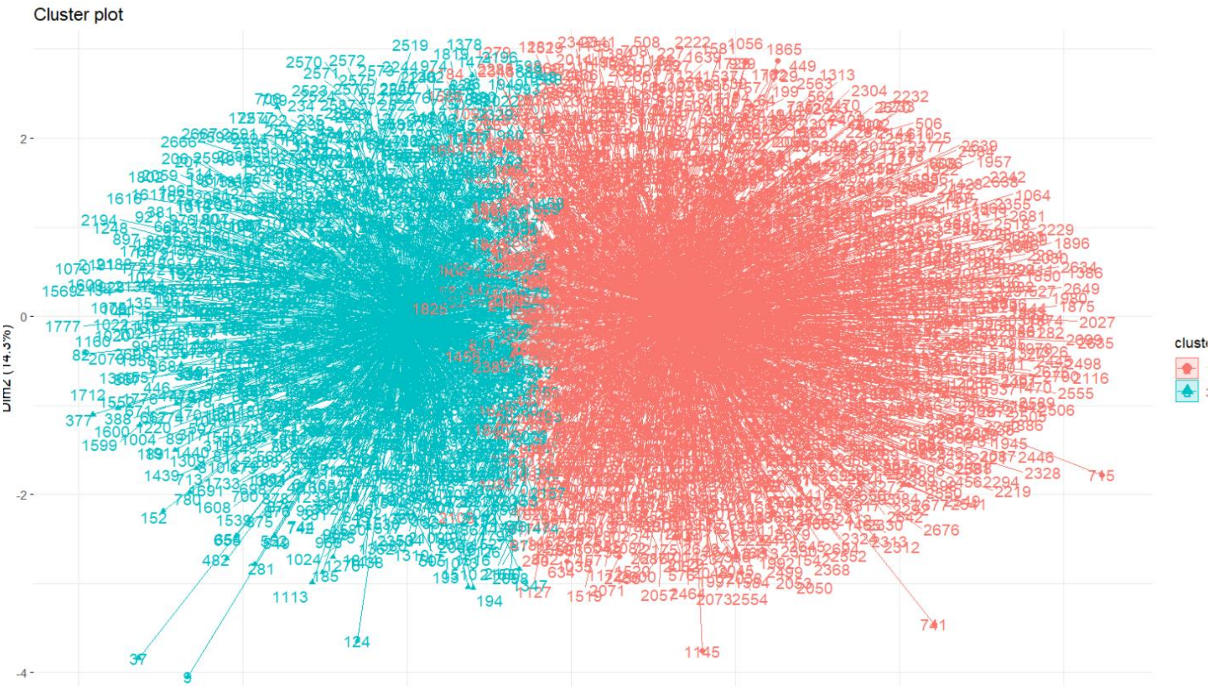
```
###################
#K after after PCA
correct_k2<-2
incorrect_k2<-5
k_kmeans2<-kmeans(final_pca, centers = correct_k2,nstart=10)

######
##visualization
set.seed(44)
fviz_cluster(k_kmeans2, data = final_pca)
##visualization 2
fviz_cluster(k_kmeans2, data = final_pca, ellipse.type = "euclid",star.plot = TRUE
####
fviz_cluster(k_kmeans2,
             data = final_pca,
             geom = "point",
             repel = TRUE,
             ggtheme = theme_minimal()) +
  theme(legend.position = "right")
```

Correct_k2=kmeans

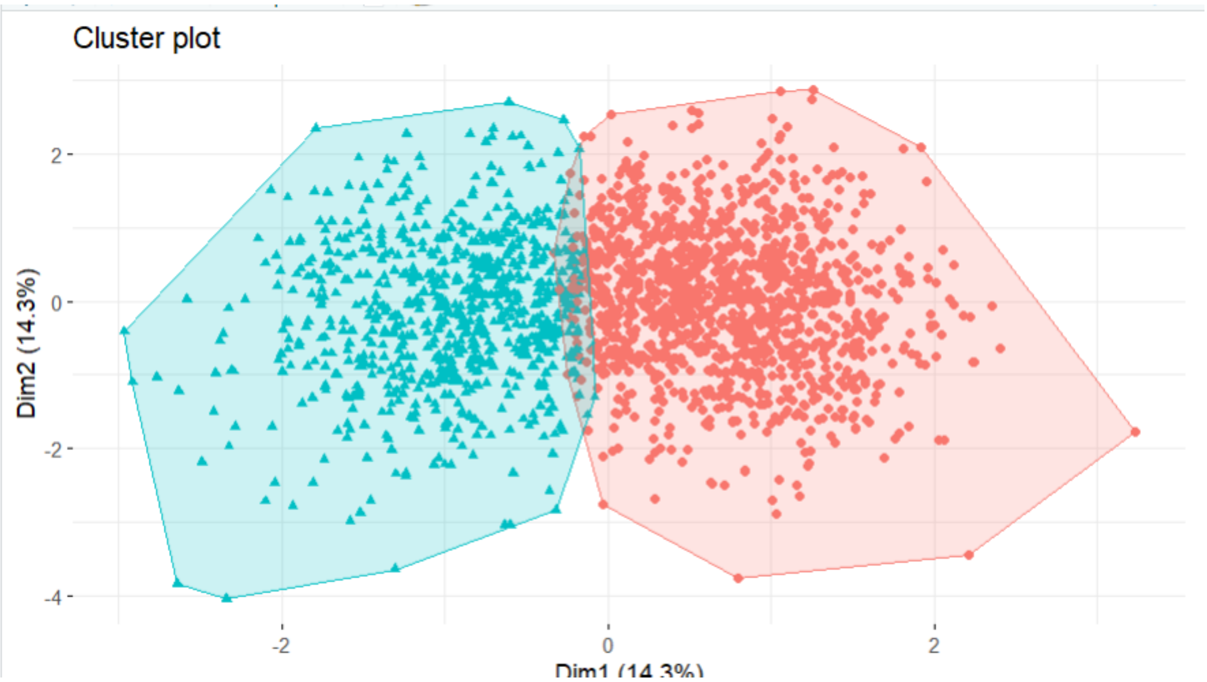
K means Analysis 1



K means Analysis 2

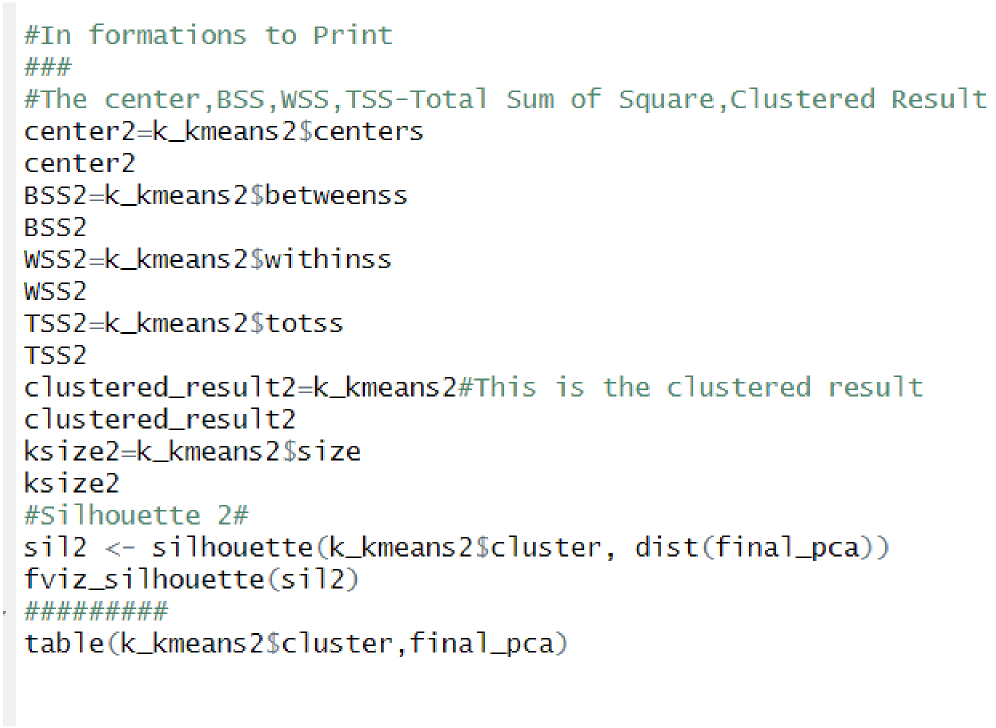


Kmeans Analysis 3:

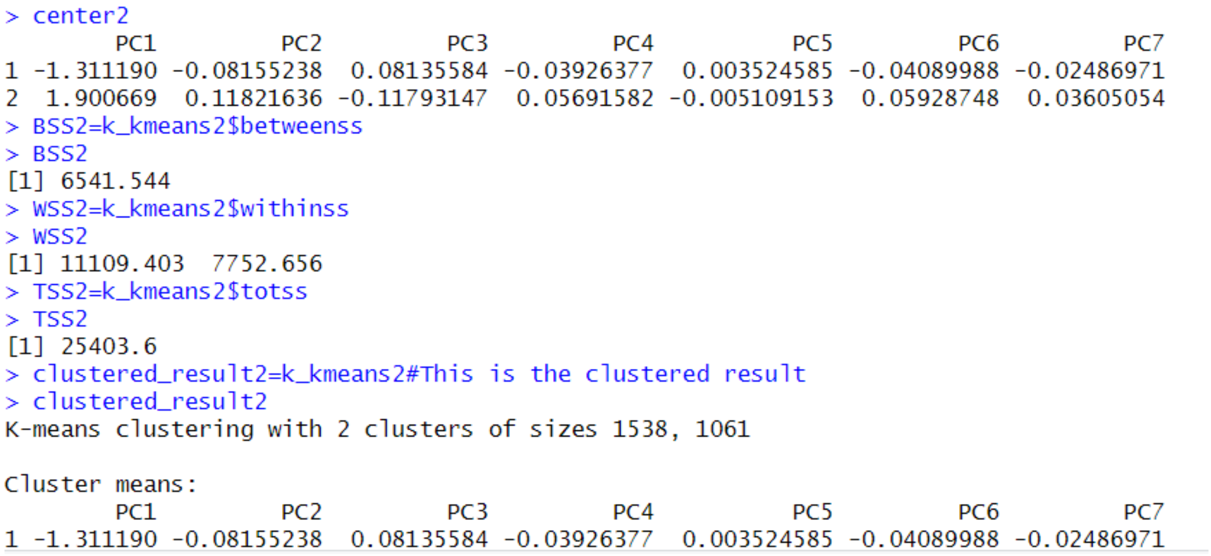


SubTask 7:

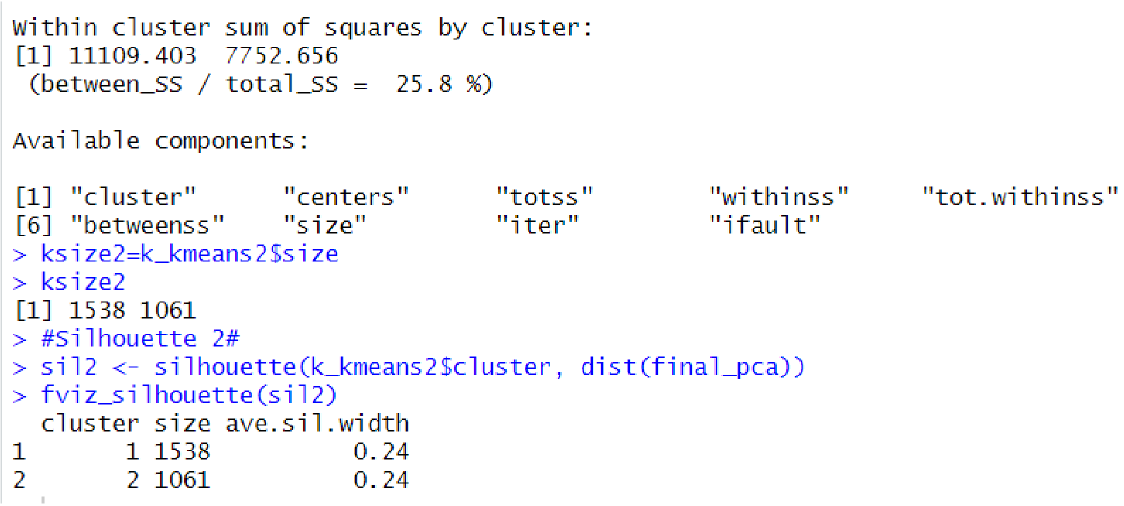
Show the silhouette plot (3 marks) and provide related discussion on this output, following this“pca-based” Kmeans attempt



```
#In formations to Print
###
#The center,BSS,WSS,TSS-Total Sum of Square,Clustered Result
center2=k_kmeans2$centers
center2
BSS2=k_kmeans2$betweenss
BSS2
WSS2=k_kmeans2$withinss
WSS2
TSS2=k_kmeans2$totss
TSS2
clustered_result2=k_kmeans2#This is the clustered result
clustered_result2
ksize2=k_kmeans2$size
ksize2
#Silhouette 2#
sil2 <- silhouette(k_kmeans2$cluster, dist(final_pca))
fviz_silhouette(sil2)
#########
table(k_kmeans2$cluster,final_pca)
```

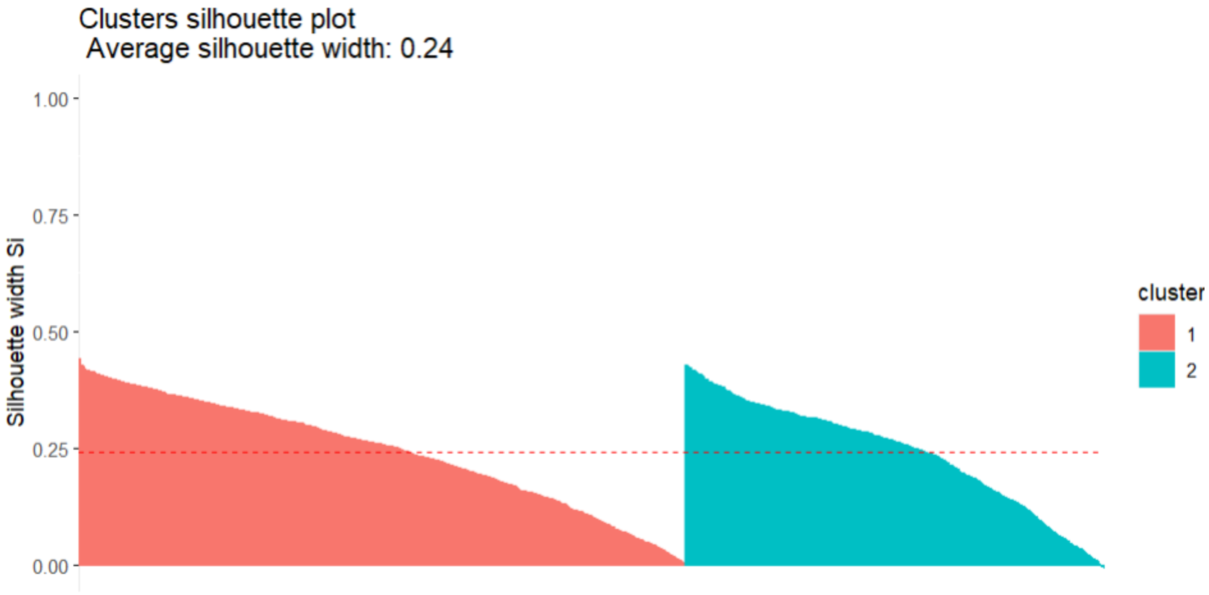


```
> center2
        PC1         PC2         PC3         PC4          PC5         PC6         PC7
1 -1.311190 -0.08155238  0.08135584 -0.03926377  0.003524585 -0.04089988 -0.02486971
2  1.900669  0.11821636 -0.11793147  0.05691582 -0.005109153  0.05928748  0.03605054
> BSS2=k_kmeans2$betweenss
> BSS2
[1] 6541.544
> WSS2=k_kmeans2$withinss
> WSS2
[1] 11109.403  7752.656
> TSS2=k_kmeans2$totss
> TSS2
[1] 25403.6
> clustered_result2=k_kmeans2#This is the clustered result
> clustered_result2
K-means clustering with 2 clusters of sizes 1538, 1061

Cluster means:
        PC1         PC2         PC3         PC4          PC5         PC6         PC7
1 -1.311190 -0.08155238  0.08135584 -0.03926377  0.003524585 -0.04089988 -0.02486971
```



```
Within cluster sum of squares by cluster:
[1] 11109.403  7752.656
 (between_SS / total_SS =  25.8 %)

Available components:

[1] "cluster"      "centers"      "totss"        "withinss"     "tot.withinss"
[6] "betweenss"    "size"         "iter"         "ifault"      
> ksize2=k_kmeans2$size
> ksize2
[1] 1538 1061
> #Silhouette 2#
> sil2 <- silhouette(k_kmeans2$cluster, dist(final_pca))
> fviz_silhouette(sil2)
  cluster size ave.sil.width
1       1 1538          0.24
2       2 1061          0.24
```

Silhouette Output after PCA:



SubTask 8:

Reference:

Wikipedia contributors. (2024,March,19). Calinski–Harabasz index. In Wikipedia. Retrieved (2024,April,26th), from https://en.wikipedia.org/wiki/Calinski%E2%80%93Harabasz_index

Implement and show the Calinski-Harabasz index. Provide, a brief discussion on the outcome of this index.

////

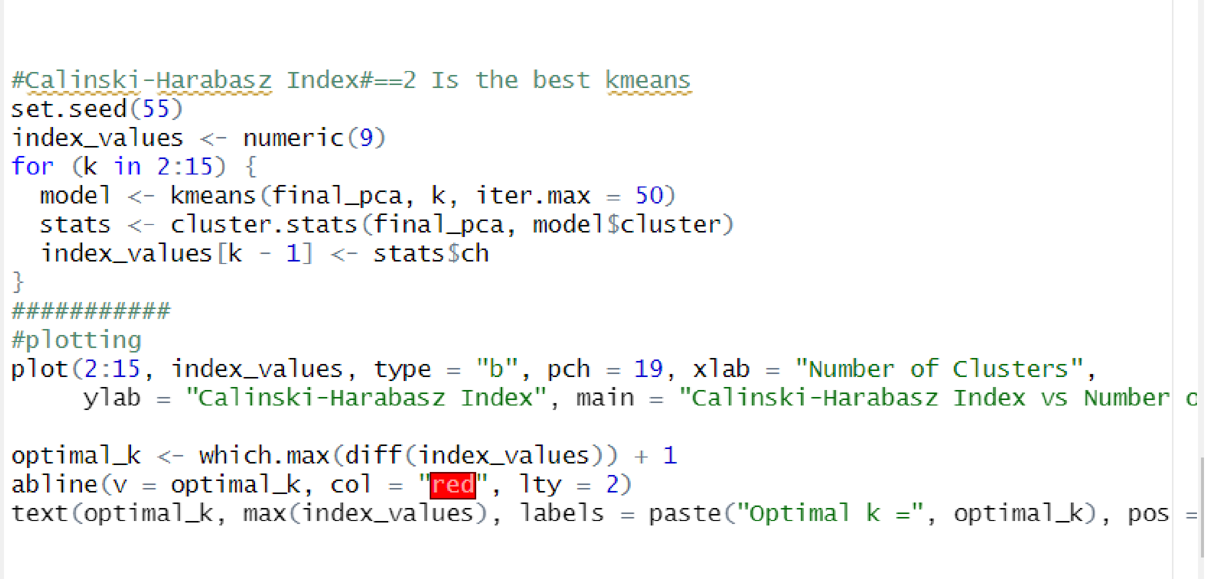
The calinski-harabasz index is a clustering technique for finding the optimal number of cluster

Procedures:

1. Perform clustering for different values of *k*.
2. Compute the CH index for each clustering result.
3. The value of *k* that yields the maximum CH index is chosen as the optimal number of clusters.

It is also defined as the ratio of BSS/WSS

CH=BSS/WSS

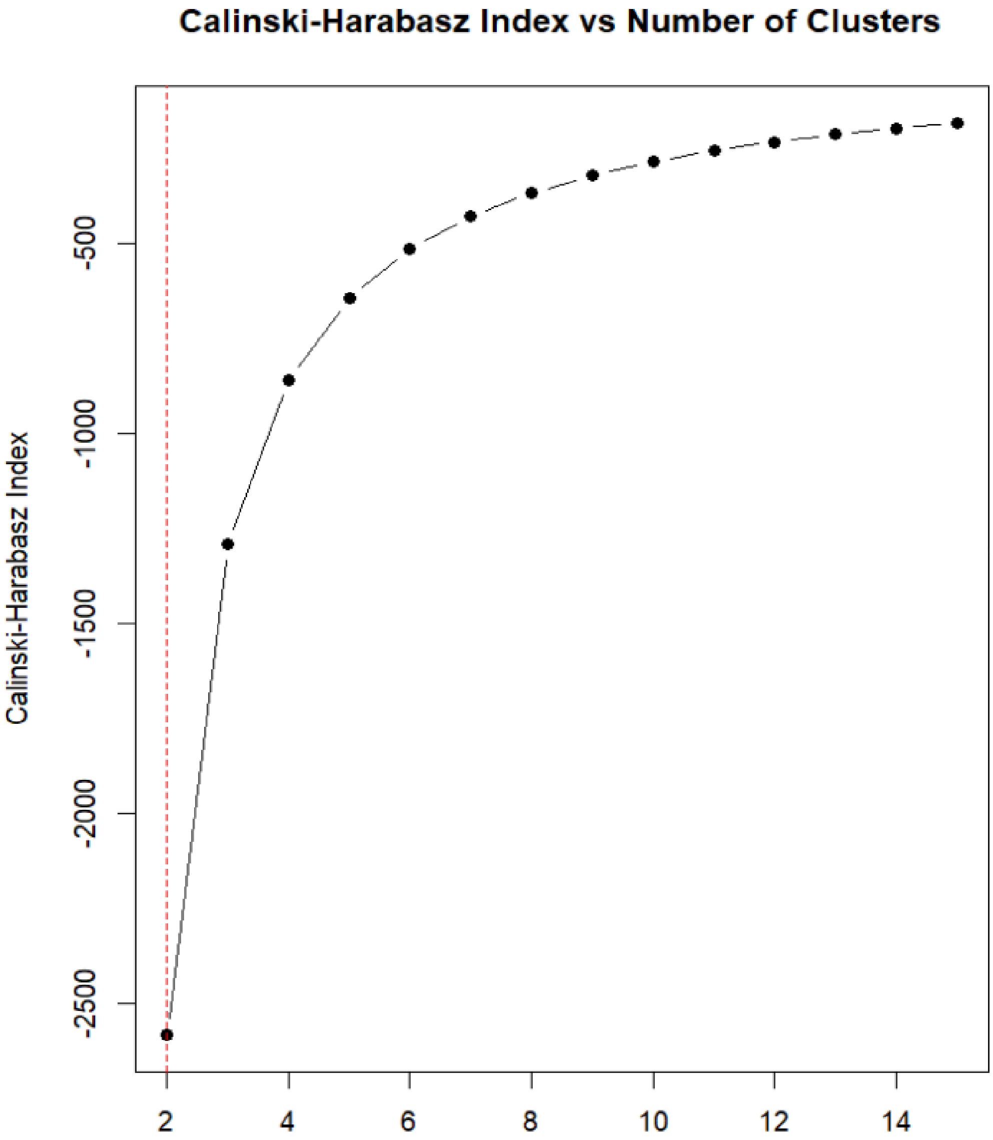


```
#Calinski-Harabasz Index#==2 Is the best kmeans
set.seed(55)
index_values <- numeric(9)
for (k in 2:15) {
  model <- kmeans(final_pca, k, iter.max = 50)
  stats <- cluster.stats(final_pca, model$cluster)
  index_values[k - 1] <- stats$ch
}
###########
#plotting
plot(2:15, index_values, type = "b", pch = 19, xlab = "Number of Clusters",
     ylab = "Calinski-Harabasz Index", main = "Calinski-Harabasz Index vs Number c

optimal_k <- which.max(diff(index_values)) + 1
abline(v = optimal_k, col = "red", lty = 2)
text(optimal_k, max(index_values), labels = paste("Optimal k =", optimal_k), pos =
```

After Implementation of the Calinski-Harabasz index

The best optimal k is 2 just like the previous techniques

Overall the best Kmeans for the data is 2 before and after PCA


Calinski-Harabasz Index vs Number of Clusters
Calinski-Harabasz Index
-500
-1000
-1500
-2000
-2500
2
4
6
8
10
12
14

CourseWork2:

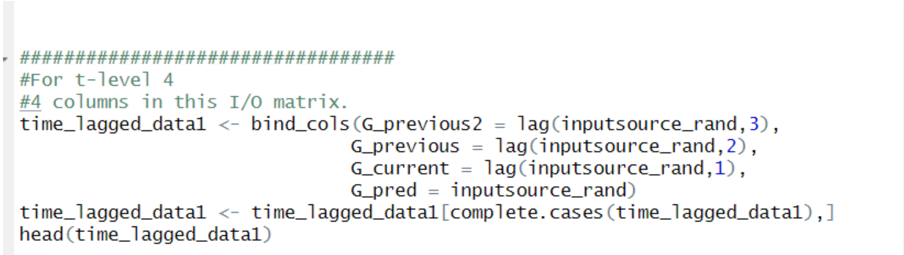
Brief discussion of the various methods used for defining the input vector in exchange rates forecasting problems 5

The method used for defining the input vector

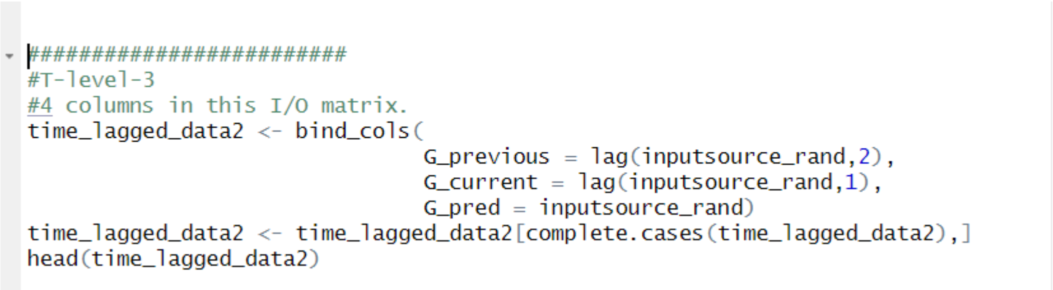
Is a time-lagged data regression model, it operates by predicting data used 3 day prior or 2 days prior as seen below and it is very efficient

2. Evidence of various adopted input vectors and the related input/output matrices for this “AR” based approach 4

Input vector of 4

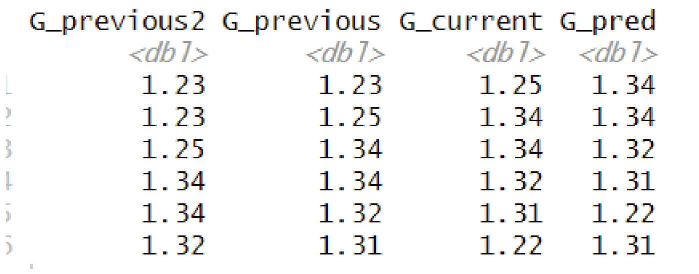


```
##################################
#For t-level 4
#4 columns in this I/O matrix.
time_lagged_data1 <- bind_cols(G_previous2 = lag(inputsource_rand,3),
                               G_previous = lag(inputsource_rand,2),
                               G_current = lag(inputsource_rand,1),
                               G_pred = inputsource_rand)
time_lagged_data1 <- time_lagged_data1[complete.cases(time_lagged_data1),]
head(time_lagged_data1)
```

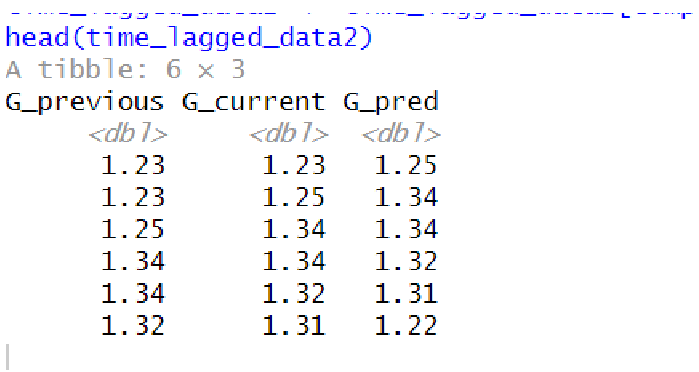


```
#########################
#T-level-3
#4 columns in this I/O matrix.
time_lagged_data2 <- bind_cols(
                              G_previous = lag(inputsource_rand,2),
                              G_current = lag(inputsource_rand,1),
                              G_pred = inputsource_rand)
time_lagged_data2 <- time_lagged_data2[complete.cases(time_lagged_data2),]
head(time_lagged_data2)
```

Outputs:



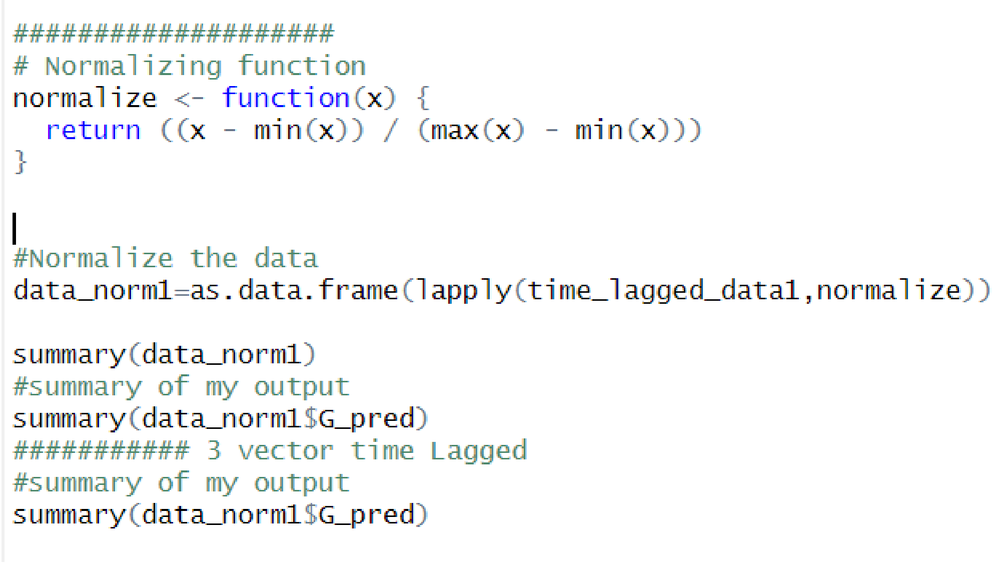
```
  G_previous2 G_previous G_current G_pred
        <dbl>      <dbl>     <dbl>  <dbl>
1        1.23       1.23      1.25   1.34
2        1.23       1.25      1.34   1.34
3        1.25       1.34      1.34   1.32
4        1.34       1.34      1.32   1.31
5        1.34       1.32      1.31   1.22
6        1.32       1.31      1.22   1.31
```



```
head(time_lagged_data2)
A tibble: 6 × 3
G_previous G_current G_pred
     <dbl>     <dbl>  <dbl>
      1.23      1.23   1.25
      1.23      1.25   1.34
      1.25      1.34   1.34
      1.34      1.34   1.32
      1.34      1.32   1.31
      1.32      1.31   1.22
```

3. Evidence of correct normalisation (3 marks) /de-normalisation (3 marks) and brief discussion of its necessity for MLP networks (3 marks) 9

T-4 normalisation



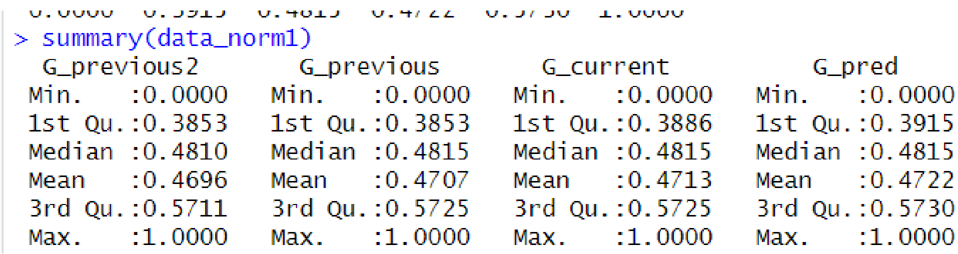
```
####################
# Normalizing function
normalize <- function(x) {
  return ((x - min(x)) / (max(x) - min(x)))
}

#Normalize the data
data_norm1=as.data.frame(lapply(time_lagged_data1,normalize))

summary(data_norm1)
#summary of my output
summary(data_norm1$G_pred)
########### 3 vector time Lagged
#summary of my output
summary(data_norm1$G_pred)
```

T-3 normalisation

```
####################
#Normalize the data
data_norm2=as.data.frame(lapply(time_lagged_data2,normalize))

summary(data_norm2)
#summary of my output
summary(data_norm2$G_pred)
########### 3 vector time Lagged
#summary of my output
summary(data_norm2$G_pred)
```

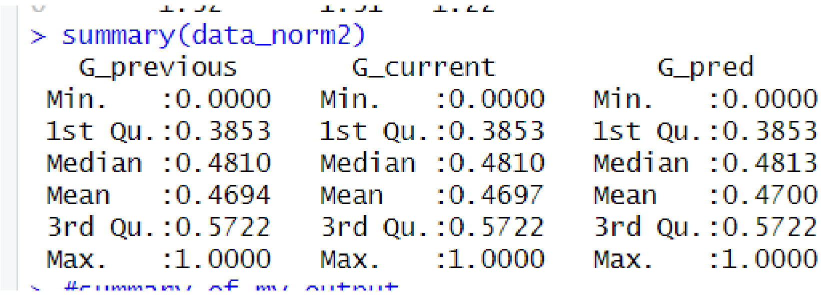
Output:

T4 normalisation:



```
 0.0000   0.3915   0.4815   0.4722   0.5730   1.0000
> summary(data_norm1)
  G_previous2        G_previous         G_current           G_pred      
 Min.   :0.0000   Min.   :0.0000   Min.   :0.0000   Min.   :0.0000  
 1st Qu.:0.3853   1st Qu.:0.3853   1st Qu.:0.3886   1st Qu.:0.3915  
 Median :0.4810   Median :0.4815   Median :0.4815   Median :0.4815  
 Mean   :0.4696   Mean   :0.4707   Mean   :0.4713   Mean   :0.4722  
 3rd Qu.:0.5711   3rd Qu.:0.5725   3rd Qu.:0.5725   3rd Qu.:0.5730  
 Max.   :1.0000   Max.   :1.0000   Max.   :1.0000   Max.   :1.0000  
```

T3 normalisation:



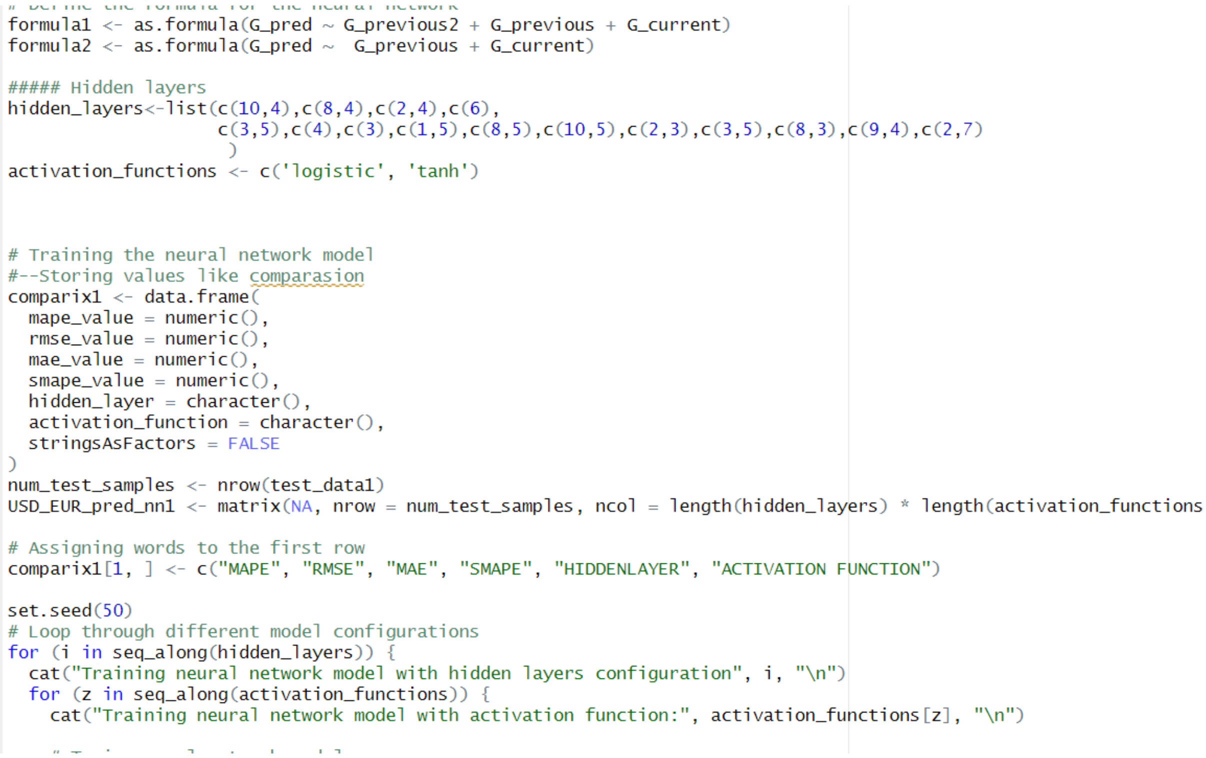
```
> summary(data_norm2)
   G_previous        G_current          G_pred      
 Min.   :0.0000   Min.   :0.0000   Min.   :0.0000  
 1st Qu.:0.3853   1st Qu.:0.3853   1st Qu.:0.3853  
 Median :0.4810   Median :0.4810   Median :0.4813  
 Mean   :0.4694   Mean   :0.4697   Mean   :0.4700  
 3rd Qu.:0.5722   3rd Qu.:0.5722   3rd Qu.:0.5722  
 Max.   :1.0000   Max.   :1.0000   Max.   :1.0000  
> #summary of my output
```

4. Implement a number of MLPs for the “AR” approach, using various internal structures(layers/nodes)/input variables/network parameters and show in the comparison,

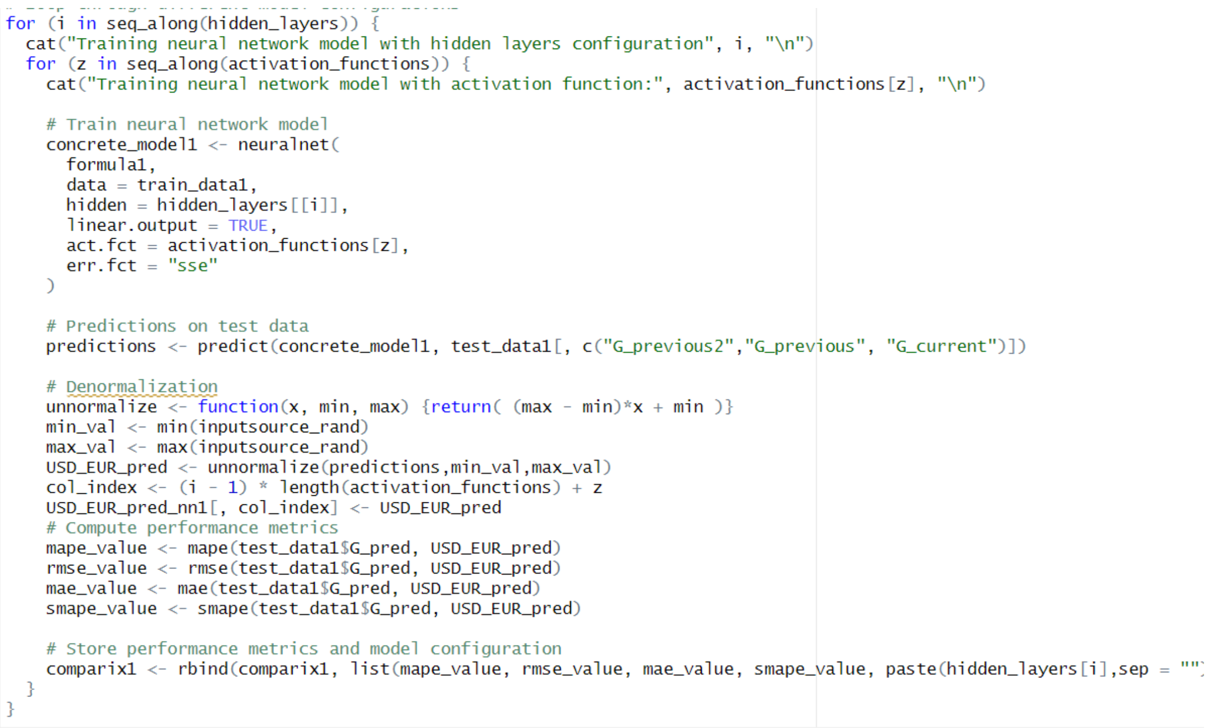
table, their performances(based on testing data) through the provided stat. indices. (4 marks for structures with different input vectors, 8 marks for different internal NN structures). 12,

6. Creation of the comparison matrix for this exchange rates case 4

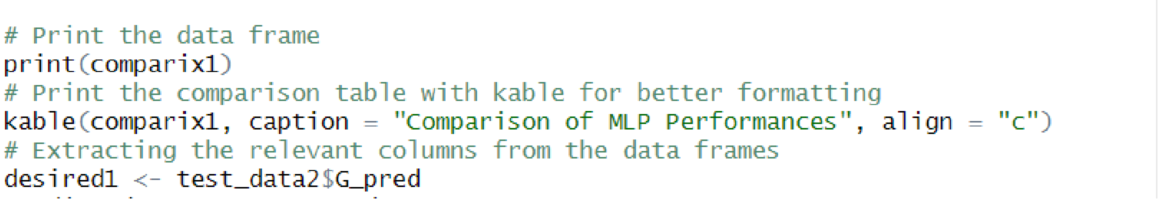
T4:



```
# Define the formula for the neural network
formula1 <- as.formula(G_pred ~ G_previous2 + G_previous + G_current)
formula2 <- as.formula(G_pred ~  G_previous + G_current)

##### Hidden layers
hidden_layers<-list(c(10,4),c(8,4),c(2,4),c(6),
                    c(3,5),c(4),c(3),c(1,5),c(8,5),c(10,5),c(2,3),c(3,5),c(8,3),c(9,4),c(2,7)
                     )
activation_functions <- c('logistic', 'tanh')



# Training the neural network model
#--Storing values like comparasion
comparix1 <- data.frame(
  mape_value = numeric(),
  rmse_value = numeric(),
  mae_value = numeric(),
  smape_value = numeric(),
  hidden_layer = character(),
  activation_function = character(),
  stringsAsFactors = FALSE
)
num_test_samples <- nrow(test_data1)
USD_EUR_pred_nn1 <- matrix(NA, nrow = num_test_samples, ncol = length(hidden_layers) * length(activation_functions

# Assigning words to the first row
comparix1[1, ] <- c("MAPE", "RMSE", "MAE", "SMAPE", "HIDDENLAYER", "ACTIVATION FUNCTION")

set.seed(50)
# Loop through different model configurations
for (i in seq_along(hidden_layers)) {
  cat("Training neural network model with hidden layers configuration", i, "\n")
  for (z in seq_along(activation_functions)) {
    cat("Training neural network model with activation function:", activation_functions[z], "\n")
```

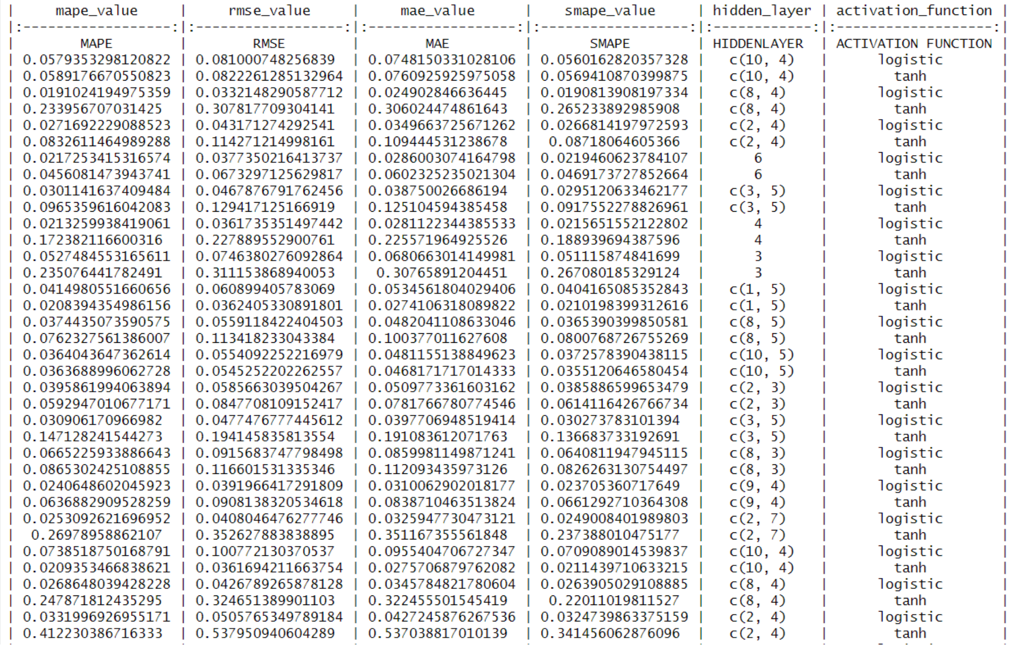


```
# Loop through different model configurations
for (i in seq_along(hidden_layers)) {
  cat("Training neural network model with hidden layers configuration", i, "\n")
  for (z in seq_along(activation_functions)) {
    cat("Training neural network model with activation function:", activation_functions[z], "\n")

    # Train neural network model
    concrete_model1 <- neuralnet(
      formula1,
      data = train_data1,
      hidden = hidden_layers[[i]],
      linear.output = TRUE,
      act.fct = activation_functions[z],
      err.fct = "sse"
    )

    # Predictions on test data
    predictions <- predict(concrete_model1, test_data1[, c("G_previous2","G_previous", "G_current")])

    # Denormalization
    unnormalize <- function(x, min, max) {return( (max - min)*x + min )}
    min_val <- min(inputsource_rand)
    max_val <- max(inputsource_rand)
    USD_EUR_pred <- unnormalize(predictions,min_val,max_val)
    col_index <- (i - 1) * length(activation_functions) + z
    USD_EUR_pred_nn1[, col_index] <- USD_EUR_pred
    # Compute performance metrics
    mape_value <- mape(test_data1$G_pred, USD_EUR_pred)
    rmse_value <- rmse(test_data1$G_pred, USD_EUR_pred)
    mae_value <- mae(test_data1$G_pred, USD_EUR_pred)
    smape_value <- smape(test_data1$G_pred, USD_EUR_pred)

    # Store performance metrics and model configuration
    comparix1 <- rbind(comparix1, list(mape_value, rmse_value, mae_value, smape_value, paste(hidden_layers[i],sep = ""
  }
}
```

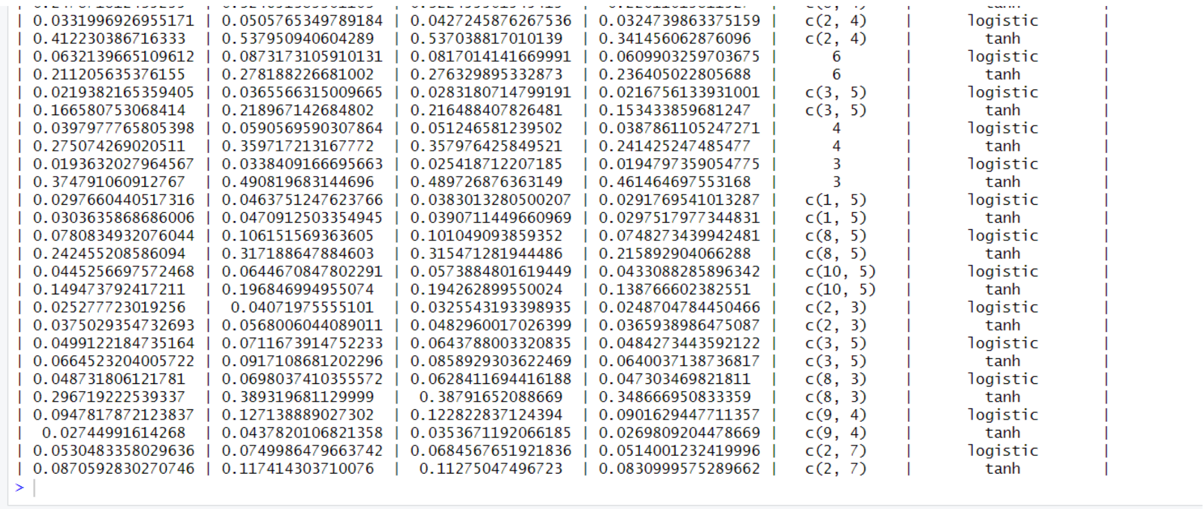


```
# Print the data frame
print(comparix1)
# Print the comparison table with kable for better formatting
kable(comparix1, caption = "Comparison of MLP Performances", align = "c")
# Extracting the relevant columns from the data frames
desired1 <- test_data2$G_pred
```

Ouput of t4:

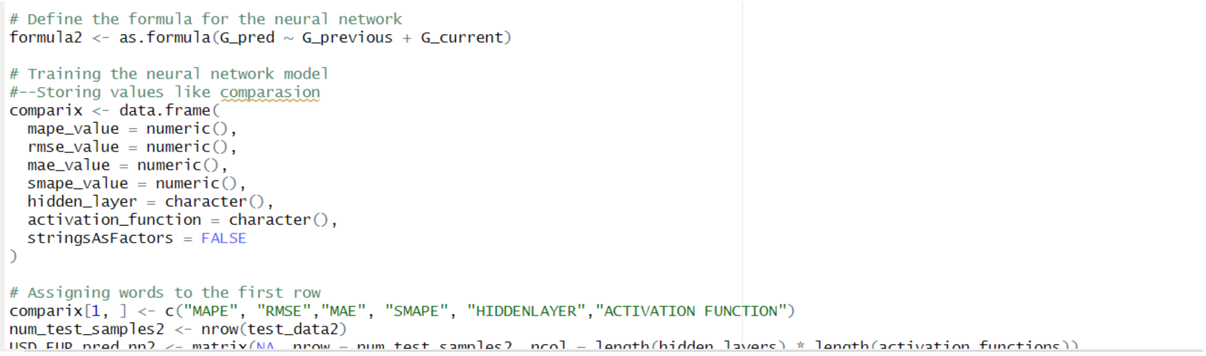


```
|     mape_value     |     rmse_value     |     mae_value      |    smape_value     | hidden_layer | activation_function |
|:------------------:|:------------------:|:------------------:|:------------------:|:------------:|:-------------------:|
|        MAPE        |        RMSE        |        MAE         |       SMAPE        | HIDDENLAYER  | ACTIVATION FUNCTION |
| 0.0579353298120822 | 0.081000748256839  | 0.0748150331028106 | 0.0560162820357328 |   c(10, 4)   |      logistic       |
| 0.0589176670550823 | 0.0822261285132964 | 0.0760925925975058 | 0.0569410870399875 |   c(10, 4)   |        tanh         |
| 0.0191024194975359 | 0.0332148290587712 | 0.024902846636445  | 0.0190813908197334 |   c(8, 4)    |      logistic       |
| 0.233956707031425  | 0.307817709304141  | 0.306024474861643  | 0.265233892985908  |   c(8, 4)    |        tanh         |
| 0.0271692229088523 | 0.043171274292541  | 0.0349663725671262 | 0.0266814197972593 |   c(2, 4)    |      logistic       |
| 0.0832611464989288 | 0.114271214998161  | 0.109444531238678  |  0.08718064605366  |   c(2, 4)    |        tanh         |
| 0.0217253415316574 | 0.0377350216413737 | 0.0286003074164798 | 0.0219460623784107 |      6       |      logistic       |
| 0.0456081473943741 | 0.0673297125629817 | 0.0602325235021304 | 0.0469173727852664 |      6       |        tanh         |
| 0.0301141637409484 | 0.0467876791762456 | 0.038750026686194  | 0.0295120633462177 |   c(3, 5)    |      logistic       |
| 0.0965359616042083 | 0.129417125166919  | 0.125104594385458  | 0.0917552278826961 |   c(3, 5)    |        tanh         |
| 0.0213259938419061 | 0.0361735351497442 | 0.0281122344385533 | 0.0215651552122802 |      4       |      logistic       |
| 0.172382116600316  | 0.227889552900761  | 0.225571964925526  | 0.188939694387596  |      4       |        tanh         |
| 0.0527484553165611 | 0.0746380276092864 | 0.0680663014149981 | 0.051115874841699  |      3       |      logistic       |
| 0.235076441782491  | 0.311153868940053  |  0.30765891204451  | 0.267080185329124  |      3       |        tanh         |
| 0.0414980551660656 | 0.060899405783069  | 0.0534561804029406 | 0.0404165085352843 |   c(1, 5)    |      logistic       |
| 0.0208394354986156 | 0.0362405330891801 | 0.0274106318089822 | 0.0210198399312616 |   c(1, 5)    |        tanh         |
| 0.0374435073590575 | 0.0559118422404503 | 0.0482041108633046 | 0.0365390399850581 |   c(8, 5)    |      logistic       |
| 0.0762327561386007 | 0.113418233043384  | 0.100377011627608  | 0.0800768726755269 |   c(8, 5)    |        tanh         |
| 0.0364043647362614 | 0.0554092252216979 | 0.0481155138849623 | 0.0372578390438115 |   c(10, 5)   |      logistic       |
| 0.0363688996062728 | 0.0545252202262557 | 0.0468171717014333 | 0.0355120646580454 |   c(10, 5)   |        tanh         |
| 0.0395861994063894 | 0.0585663039504267 | 0.0509773361603162 | 0.0385886599653479 |   c(2, 3)    |      logistic       |
| 0.0592947010677171 | 0.0847708109152417 | 0.0781766780774546 | 0.0614116426766734 |   c(2, 3)    |        tanh         |
| 0.030906170966982  | 0.0477476777445612 | 0.0397706948519414 | 0.030273783101394  |   c(3, 5)    |      logistic       |
| 0.147128241544273  | 0.194145835813554  | 0.191083612071763  | 0.136683733192691  |   c(3, 5)    |        tanh         |
| 0.0665225933886643 | 0.0915683747798498 | 0.0859981149871241 | 0.0640811947945115 |   c(8, 3)    |      logistic       |
| 0.0865302425108855 | 0.116601531335346  | 0.112093435973126  | 0.0826263130754497 |   c(8, 3)    |        tanh         |
| 0.0240648602045923 | 0.0391966417291809 | 0.0310062902018177 | 0.023705360717649  |   c(9, 4)    |      logistic       |
| 0.0636882909528259 | 0.0908138320534618 | 0.0838710463513824 | 0.0661292710364308 |   c(9, 4)    |        tanh         |
| 0.0253092621696952 | 0.0408046476277746 | 0.0325947730473121 | 0.0249008401989803 |   c(2, 7)    |      logistic       |
|  0.26978958862107  | 0.352627883838895  | 0.351167355561848  | 0.237388010475177  |   c(2, 7)    |        tanh         |
| 0.0738518750168791 | 0.100772130370537  | 0.0955404706727347 | 0.0709089014539837 |   c(10, 4)   |      logistic       |
| 0.0209353466838621 | 0.0361694211663754 | 0.0275706879762082 | 0.0211439710633215 |   c(10, 4)   |        tanh         |
| 0.0268648039428228 | 0.0426789265878128 | 0.0345784821780604 | 0.0263905029108885 |   c(8, 4)    |      logistic       |
| 0.247871812435295  | 0.324651389901103  | 0.322455501545419  |  0.22011019811527  |   c(8, 4)    |        tanh         |
| 0.0331996926955171 | 0.0505765349789184 | 0.0427245876267536 | 0.0324739863375159 |   c(2, 4)    |      logistic       |
| 0.412230386716333  | 0.537950940604289  | 0.537038817010139  | 0.341456062876096  |   c(2, 4)    |        tanh         |
```

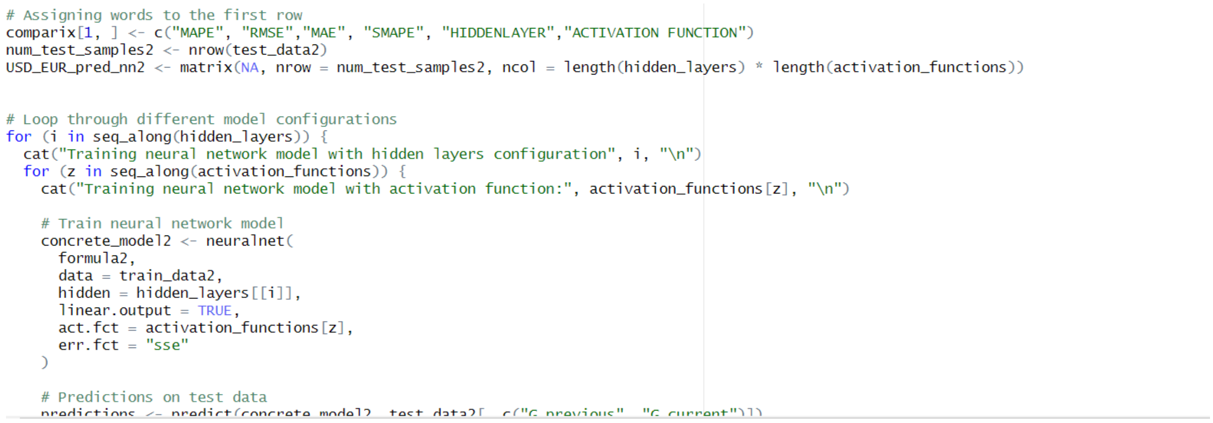


```
| 0.0331996926955171 | 0.0505765349789184 | 0.0427245876267536 | 0.0324739863375159 |  c(2, 4)  |     logistic     |
| 0.412230386716333  | 0.537950940604289  | 0.537038817010139  | 0.341456062876096  |  c(2, 4)  |       tanh       |
| 0.0632139665109612 | 0.0873173105910131 | 0.0817014141669991 | 0.0609903259703675 |     6     |     logistic     |
| 0.211205635376155  | 0.278188226681002  | 0.276329895332873  | 0.236405022805688  |     6     |       tanh       |
| 0.0219382165359405 | 0.0365566315009665 | 0.0283180714799191 | 0.0216756133931001 |  c(3, 5)  |     logistic     |
| 0.166580753068414  | 0.218967142684802  | 0.216488407826481  | 0.153433859681247  |  c(3, 5)  |       tanh       |
| 0.0397977765805398 | 0.0590569590307864 | 0.051246581239502  | 0.0387861105247271 |     4     |     logistic     |
| 0.275074269020511  | 0.359717213167772  | 0.357976425849521  | 0.241425247485477  |     4     |       tanh       |
| 0.0193632027964567 | 0.0338409166695663 | 0.025418712207185  | 0.0194797359054775 |     3     |     logistic     |
| 0.374791060912767  | 0.490819683144696  | 0.489726876363149  | 0.461464697553168  |     3     |       tanh       |
| 0.0297660440517316 | 0.0463751247623766 | 0.0383013280500207 | 0.0291769541013287 |  c(1, 5)  |     logistic     |
| 0.0303635868686006 | 0.0470912503354945 | 0.0390711449660969 | 0.0297517977344831 |  c(1, 5)  |       tanh       |
| 0.0780834932076044 | 0.106151569363605  | 0.101049093859352  | 0.0748273439942481 |  c(8, 5)  |     logistic     |
| 0.242455208586094  | 0.317188647884603  | 0.315471281944486  | 0.215892904066288  |  c(8, 5)  |       tanh       |
| 0.0445256697572468 | 0.0644670847802291 | 0.0573884801619449 | 0.0433088285896342 |  c(10, 5) |     logistic     |
| 0.149473792417211  | 0.196846994955074  | 0.194262899550024  | 0.138766602382551  |  c(10, 5) |       tanh       |
| 0.025277723019256  |  0.04071975555101  | 0.0325543193398935 | 0.0248704784450466 |  c(2, 3)  |     logistic     |
| 0.0375029354732693 | 0.0568006044089011 | 0.0482960017026399 | 0.0365938986475087 |  c(2, 3)  |       tanh       |
| 0.0499122184735164 | 0.0711673914752233 | 0.0643788003320835 | 0.0484273443592122 |  c(3, 5)  |     logistic     |
| 0.0664523204005722 | 0.0917108681202296 | 0.0858929303622469 | 0.0640037138736817 |  c(3, 5)  |       tanh       |
| 0.048731806121781  | 0.0698037410355572 | 0.0628411694416188 | 0.047303469821811  |  c(8, 3)  |     logistic     |
| 0.296719222539337  | 0.389319681129999  |  0.38791652088669  | 0.348666950833359  |  c(8, 3)  |       tanh       |
| 0.0947817872123837 | 0.127138889027302  | 0.122822837124394  | 0.0901629447711357 |  c(9, 4)  |     logistic     |
|  0.02744991614268  | 0.0437820106821358 | 0.0353671192066185 | 0.0269809204478669 |  c(9, 4)  |       tanh       |
| 0.0530483358029636 | 0.0749986479663742 | 0.0684567651921836 | 0.0514001232419996 |  c(2, 7)  |     logistic     |
| 0.0870592830270746 | 0.117414303710076  |  0.11275047496723  | 0.0830999575289662 |  c(2, 7)  |       tanh       |
> 
```

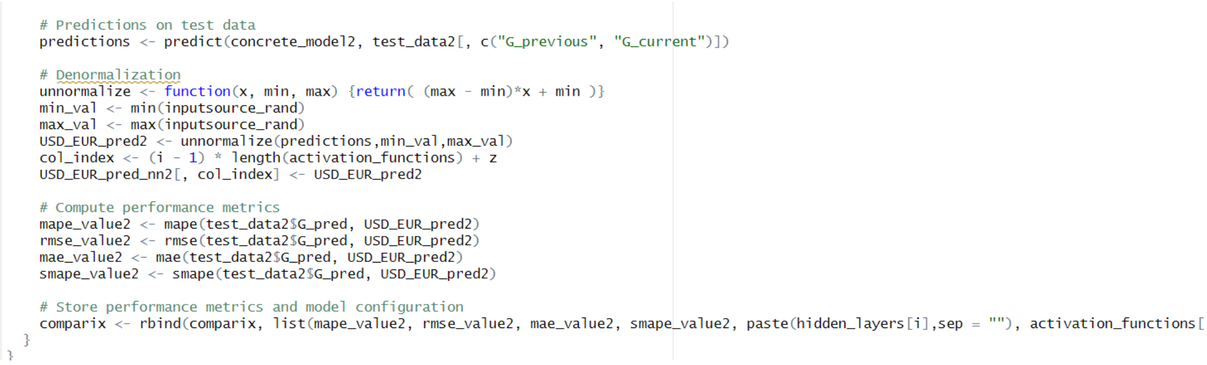
Tlevel3:



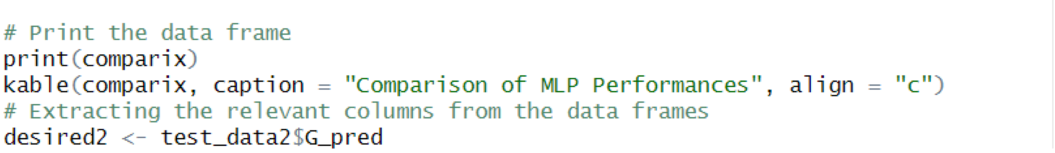
```
# Define the formula for the neural network
formula2 <- as.formula(G_pred ~ G_previous + G_current)

# Training the neural network model
#--Storing values like comparasion
comparix <- data.frame(
  mape_value = numeric(),
  rmse_value = numeric(),
  mae_value = numeric(),
  smape_value = numeric(),
  hidden_layer = character(),
  activation_function = character(),
  stringsAsFactors = FALSE
)

# Assigning words to the first row
comparix[1, ] <- c("MAPE", "RMSE","MAE", "SMAPE", "HIDDENLAYER","ACTIVATION FUNCTION")
num_test_samples2 <- nrow(test_data2)
USD_EUR_pred_nn2 <- matrix(NA, nrow = num_test_samples2, ncol = length(hidden_layers) * length(activation_functions))
```



```
# Assigning words to the first row
comparix[1, ] <- c("MAPE", "RMSE","MAE", "SMAPE", "HIDDENLAYER","ACTIVATION FUNCTION")
num_test_samples2 <- nrow(test_data2)
USD_EUR_pred_nn2 <- matrix(NA, nrow = num_test_samples2, ncol = length(hidden_layers) * length(activation_functions))


# Loop through different model configurations
for (i in seq_along(hidden_layers)) {
  cat("Training neural network model with hidden layers configuration", i, "\n")
  for (z in seq_along(activation_functions)) {
    cat("Training neural network model with activation function:", activation_functions[z], "\n")

    # Train neural network model
    concrete_model2 <- neuralnet(
      formula2,
      data = train_data2,
      hidden = hidden_layers[[i]],
      linear.output = TRUE,
      act.fct = activation_functions[z],
      err.fct = "sse"
    )

    # Predictions on test data
    predictions <- predict(concrete_model2, test_data2[, c("G_previous", "G_current")])
```



```
    # Predictions on test data
    predictions <- predict(concrete_model2, test_data2[, c("G_previous", "G_current")])

    # Denormalization
    unnormalize <- function(x, min, max) {return( (max - min)*x + min )}
    min_val <- min(inputsource_rand)
    max_val <- max(inputsource_rand)
    USD_EUR_pred2 <- unnormalize(predictions,min_val,max_val)
    col_index <- (i - 1) * length(activation_functions) + z
    USD_EUR_pred_nn2[, col_index] <- USD_EUR_pred2

    # Compute performance metrics
    mape_value2 <- mape(test_data2$G_pred, USD_EUR_pred2)
    rmse_value2 <- rmse(test_data2$G_pred, USD_EUR_pred2)
    mae_value2 <- mae(test_data2$G_pred, USD_EUR_pred2)
    smape_value2 <- smape(test_data2$G_pred, USD_EUR_pred2)

    # Store performance metrics and model configuration
    comparix <- rbind(comparix, list(mape_value2, rmse_value2, mae_value2, smape_value2, paste(hidden_layers[i],sep = ""), activation_functions[
  }
}
```



```
# Print the data frame
print(comparix)
kable(comparix, caption = "Comparison of MLP Performances", align = "c")
# Extracting the relevant columns from the data frames
desired2 <- test_data2$G_pred
```

4. Discussion of the meaning of these four stat. indices (2 marks for each index) 8

MAPE (Mean Absolute Percentage Error): MAPE measures the average magnitude of errors as a percentage of the actual values. It provides the average percentage difference between predicted and actual values. It is calculated as the average of the absolute percentage differences between predicted and actual values.

RMSE (Root Mean Squared Error): RMSE measures the average magnitude of the errors between predicted and actual values. It represents the square root of the average of the squared differences between predicted and actual values. RMSE gives more weight to larger errors.

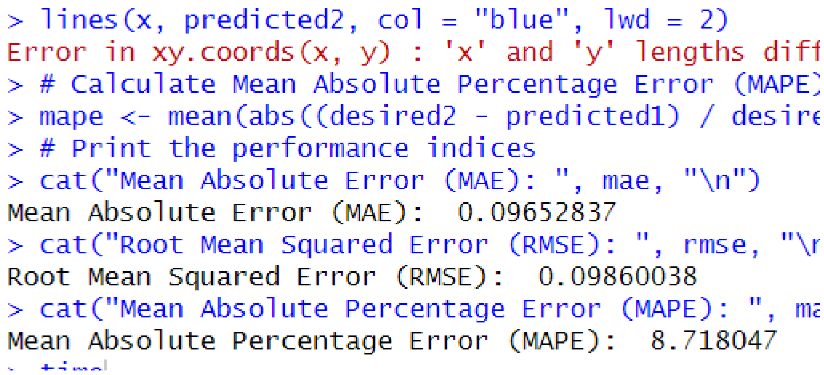
MAE (Mean Absolute Error): MAE measures the average magnitude of errors between predicted and actual values. It represents the average of the absolute differences between predicted and actual values. MAE is less sensitive to outliers compared to RMSE.

SMAPE (Symmetric Mean Absolute Percentage Error): SMAPE is similar to MAPE but considers the relative error symmetrically around zero. It calculates the average of the absolute percentage differences between predicted and actual values relative to their average. SMAPE is commonly used in forecasting tasks.

7. Discuss the issue of “efficiency” with your two best NN structures 4

T4:

The best layer is c(8,4) logistics=>

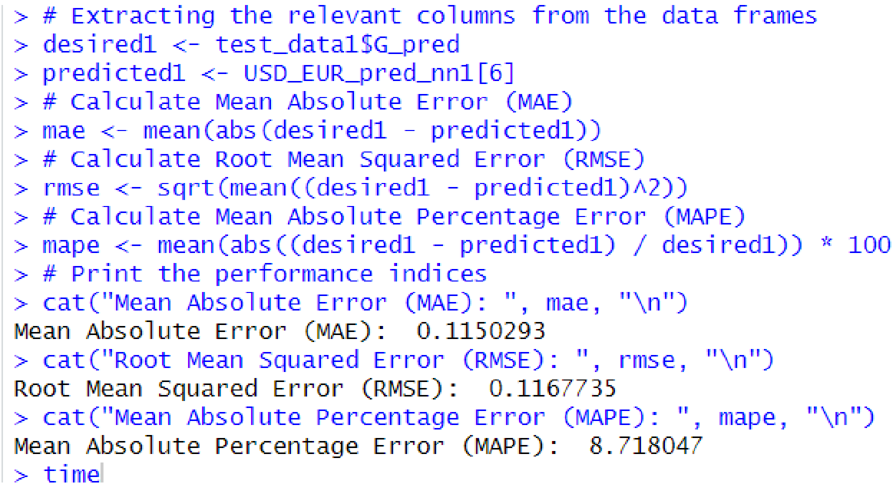


```
> lines(x, predicted2, col = "blue", lwd = 2)
Error in xy.coords(x, y) : 'x' and 'y' lengths diff
> # Calculate Mean Absolute Percentage Error (MAPE
> mape <- mean(abs((desired2 - predicted1) / desire
> # Print the performance indices
> cat("Mean Absolute Error (MAE): ", mae, "\n")
Mean Absolute Error (MAE):  0.09652837
> cat("Root Mean Squared Error (RMSE): ", rmse, "\r
Root Mean Squared Error (RMSE):  0.09860038
> cat("Mean Absolute Percentage Error (MAPE): ", ma
Mean Absolute Percentage Error (MAPE):  8.718047
```

2)C(2,4) logistics;

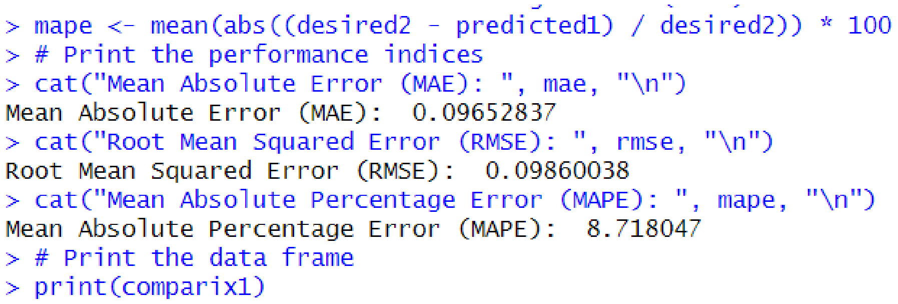
T3)

Same for t-4 level and t3 levels

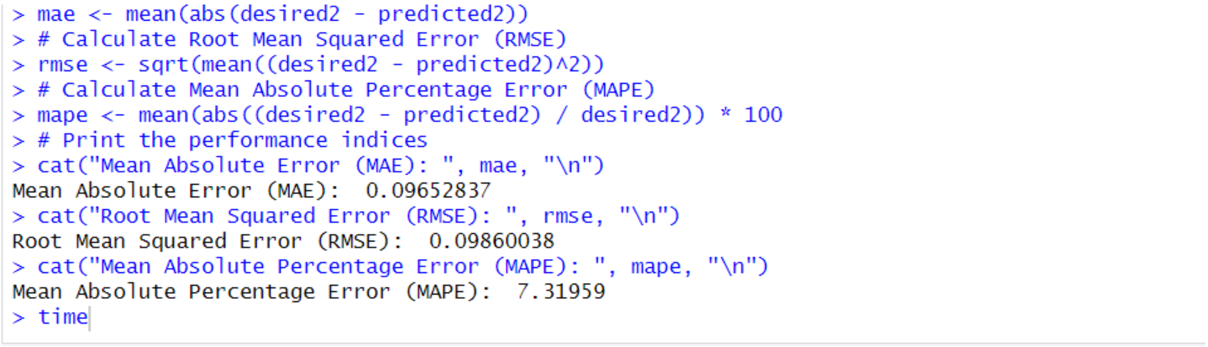


```
> # Extracting the relevant columns from the data frames
> desired1 <- test_data1$G_pred
> predicted1 <- USD_EUR_pred_nn1[6]
> # Calculate Mean Absolute Error (MAE)
> mae <- mean(abs(desired1 - predicted1))
> # Calculate Root Mean Squared Error (RMSE)
> rmse <- sqrt(mean((desired1 - predicted1)^2))
> # Calculate Mean Absolute Percentage Error (MAPE)
> mape <- mean(abs((desired1 - predicted1) / desired1)) * 100
> # Print the performance indices
> cat("Mean Absolute Error (MAE): ", mae, "\n")
Mean Absolute Error (MAE):  0.1150293
> cat("Root Mean Squared Error (RMSE): ", rmse, "\n")
Root Mean Squared Error (RMSE):  0.1167735
> cat("Mean Absolute Percentage Error (MAPE): ", mape, "\n")
Mean Absolute Percentage Error (MAPE):  8.718047
> time
```

8. Provide your best results both graphically (your prediction output vs. desired output)and via performance indices (2 marks for the graphical display and 2 marks for showing he requested statistical indices)



```
> mape <- mean(abs((desired2 - predicted1) / desired2)) * 100
> # Print the performance indices
> cat("Mean Absolute Error (MAE): ", mae, "\n")
Mean Absolute Error (MAE):  0.09652837
> cat("Root Mean Squared Error (RMSE): ", rmse, "\n")
Root Mean Squared Error (RMSE):  0.09860038
> cat("Mean Absolute Percentage Error (MAPE): ", mape, "\n")
Mean Absolute Percentage Error (MAPE):  8.718047
> # Print the data frame
> print(comparix1)
```



```
> mae <- mean(abs(desired2 - predicted2))
> # Calculate Root Mean Squared Error (RMSE)
> rmse <- sqrt(mean((desired2 - predicted2)^2))
> # Calculate Mean Absolute Percentage Error (MAPE)
> mape <- mean(abs((desired2 - predicted2) / desired2)) * 100
> # Print the performance indices
> cat("Mean Absolute Error (MAE): ", mae, "\n")
Mean Absolute Error (MAE):  0.09652837
> cat("Root Mean Squared Error (RMSE): ", rmse, "\n")
Root Mean Squared Error (RMSE):  0.09860038
> cat("Mean Absolute Percentage Error (MAPE): ", mape, "\n")
Mean Absolute Percentage Error (MAPE):  7.31959
> time
```